Requested Patent: JP10177498A

Title: MEMORY MANAGEMENT OF FAULT TOLERANT COMPUTER SYSTEM ;

Abstracted Patent: JP10177498 ;

Publication Date: 1998-06-30 ;

Inventor(s): WILLIAMS EMRYS JOHN ;

Applicant(s): SUN MICROSYSTEMS INC ;

4/6 JAPIO - (C) JPO- image
PN - JP 10177498 A 19980630 [JP10177498]
TI - ***MEMORY*** ***MANAGEMENT*** OF ***FAULT*** ***TOLERANT*** COMPUTER SYSTEM
IN - WILLIAMS EMRYS JOHN
PA - SUN MICROSYST INC
AP - JP21141597 19970630 [1997JP-0211415]
PR - US96 675265 19960701 [1996US-0675265]
IC1 - G06F-011/18
AB - PROBLEM TO BE SOLVED: To provide an automatic and prompt method to recover from a small step-out event.
- SOLUTION: A ***memory*** ***management*** system for a ***fault*** ***tolerant*** computer system includes a first recording mechanism 25 capable of recording the ***memory*** updating (writing) event, a second recording mechanism 26 provided with capacity to record the ***memory*** updating event, a fault input for a fault signal to activate the first recording mechanism in the case of the fault (step-out) event and a ***memory*** reintegration mechanism 27 at least to reintegrate the ***memory*** of the part which discriminated by the first and the second recording mechanisms. Since only comparatively small number of locations are corrected in both ***memory*** systems of processing sets of step-out and in operation, recovery from the small step-out between the processing sets in a system of lock-step method is promptly and effectively achieved by copying a ***memory*** page discriminated by the first and the second recording mechanisms from the processing set in operation to the step-out processing set.
-

5/6 JAPIO - (C) JPO- image
PN - JP 08137814 A 19960531 [JP08137814]
TI - ***FAULT*** ***TOLERANT*** SYSTEM
IN - KANAZAWA YUJI
PA - FUJITSU LTD
AP - JP27519594 19941109 [1994JP-0275195]
IC1 - G06F-015/16

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号

特開平10－177498

(43)公開日　平成10年(1998)６月30日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　11/18	３１０	Ｇ０６Ｆ　11/18	３１０Ｇ

審査請求　未請求　請求項の数41　ＦＤ　外国語出願　（全 43 頁）

(21)出願番号　特願平9－211415

(22)出願日　平成９年(1997)６月30日

(31)優先権主張番号　０８／６７５２６５

(32)優先日　1996年７月１日

(33)優先権主張国　米国（US）

(71)出願人　591064003
サン・マイクロシステムズ・インコーポレーテッド
SUN MICROSYSTEMS, INCORPORATED
アメリカ合衆国 94303 カリフォルニア州・パロ アルト・サン アントニオ ロード・901

(74)代理人　弁理士　山川　政樹

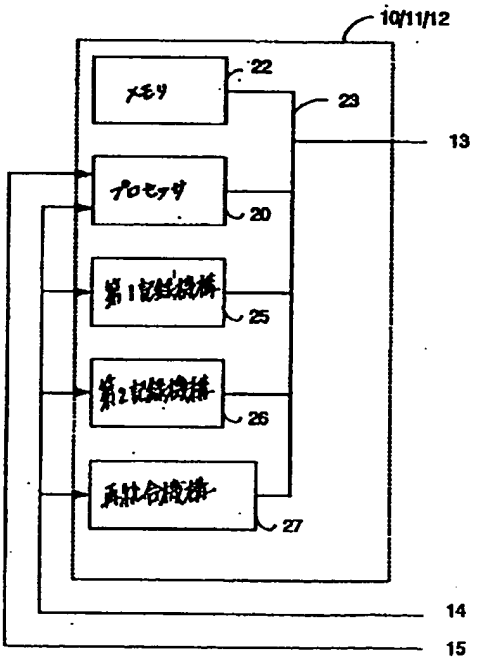
最終頁に続く

(54)【発明の名称】　フォルトトレラント・コンピュータ・システムのメモリ・マネージメント

(57)【要約】

【課題】　小規模な同期はずれイベントから回復する自動的かつ迅速な方法を提供すること。

【解決手段】　フォールト・トレラント・コンピュータ・システム用のメモリ管理システムは、メモリ更新（書込み）イベントを記録することができる第１記録機構（２５）と、メモリ更新イベントを記録するだけの容量を備えた第２記録機構（２６）と、障害（同期はずれ）イベントの場合に第１記録機構を活動化するための障害信号用の障害入力と、少なくとも第１および第２記録機構で識別された部分のメモリを再統合するメモリ再統合機構（２７）とを含む。ロックステップ方式のシステム中の処理セット間の小規模な同期はずれからの回復は、同期はずれまたは動作中の処理セットのどちらのメモリ・システム内でも、比較的少数のロケーションのみが修正されているはずなので、第１および第２記録機構で識別されたメモリ・ページを、動作中の処理セットから同期はずれ処理セットにコピーすることによって、迅速かつ効果的に達成することができる。



【特許請求の範囲】

【請求項１】　活動化されてメモリ更新イベントを記録することができる第１記録機構と、少なくとも限られた数のメモリ更新イベントを記録するだけの容量を備えた第２記録機構と、障害イベントの場合に前記第１記録機構を活動化するための障害信号用の障害入力と、少なくとも前記第１および第２記録機構で識別された部分のメモリを再統合するメモリ再統合機構とを含む、フォールト・トレラント・コンピュータ・システム用のメモリ管理システム。

【請求項２】　前記第１記録機構が、複数のメモリ・ページのそれぞれに対するエントリと、前記第１記録機構が活動化されてページが書込みされるたびに、そのページのエントリに書込みされるコードとを備えた記憶装置を含むメモリ管理ユニットであることを特徴とする、請求項１に記載のメモリ管理システム。

【請求項３】　前記第１記録機構が、前記第１記録機構を活動化するための障害信号を受領するように接続されたイネーブル入力を有することを特徴とする、請求項２に記載のメモリ管理システム。

【請求項４】　前記第２記録機構が、障害イベントに続いて前記第１記録機構を活動化する時間をカバーするのに十分な数まで、最近のメモリ更新イベントの記録を維持することを特徴とする、請求項１に記載のメモリ管理システム。

【請求項５】　前記第２記録機構が先入れ先出しバッファを含むことを特徴とする、請求項４に記載のメモリ管理システム。

【請求項６】　前記第１記録機構が前記先入れ先出しバッファの出力に接続されていることを特徴とする、請求項５に記載のメモリ管理システム。

【請求項７】　前記先入れ先出しバッファが、更新アドレスを所定数まで格納し、アドレス・デコーダが、前記先入れ先出しバッファの前記出力に接続されて、前記先入れ先出しバッファから出力されるメモリ更新アドレスを表すページ信号を生成し、前記アドレス・デコーダが、前記障害信号に応答して、前記ページ信号を前記第１記録機構に渡すことを特徴とする、請求項６に記載のメモリ管理システム。

【請求項８】　前記第２記録機構が論理解析器を含むことを特徴とする、請求項１に記載のメモリ管理システム。

【請求項９】　前記第２記録機構が、障害イベントに続いて前記第１記録機構を活動化する時間をカバーするのに十分な数まで、最近のメモリ更新イベントの記録を維持し、前記第２記録手段の動作が、前記障害信号に応答して阻止されることを特徴とする、請求項１に記載のメモリ管理システム。

【請求項１０】　前記第１記録機構がソフトウェアが生成する更新イベントのリストを有することを特徴とする、請求項９に記載のメモリ管理システム。

【請求項１１】　前記第２記録機構が、主メモリ中で維持されるアドレス変換バッファおよびメモリ・アクセス表を含むことを特徴とする、請求項１に記載のメモリ管理システム。

【請求項１２】　前記メモリ再統合機構が、前記第１および第２記録機構で識別されたメモリ・ページを再統合するように動作可能であることを特徴とする、請求項１に記載のメモリ管理システム。

【請求項１３】　個々にプロセッサおよび内部メモリを含み、ロックステップ方式で動作する複数の同期処理セットと、同期はずれイベントを検出し、同期はずれ信号を生成する同期はずれ検出器とを含むシステムであって、個々の処理セットが請求項１に記載のメモリ管理システムを含むことを特徴とする、フォールト・トレラント・コンピュータ・システム。

【請求項１４】　個々に内部メモリを備えたプロセッサを含み、ロックステップ方式で動作する複数の同期処理セットと、同期はずれイベントを検出し、同期はずれ信号を生成する同期はずれ検出器とを含むシステムであって、個々の処理セットがまた、活動化されてメモリ書込みイベントを記録することができる第１記録機構と、少なくとも限られた数のメモリ書込みイベントを記録するだけの容量を備えた第２記録機構と、同期はずれイベントの場合に、同期はずれ信号を受領して前記第１記録機構を活動化する障害入力と、少なくとも第１および第２記録機構で識別された部分のメモリを、同期はずれ処理セット中で再統合するメモリ再統合機構とを含むことを特徴とする、フォールト・トレラント・コンピュータ・システム。

【請求項１５】　前記第１記録機構が、複数のメモリ・ページのそれぞれに対するエントリと、前記第１記録機構が活動化されてページが書込みされるたびに、そのページのエントリに書込みされるコードとを備えたＲＡＭを含むメモリ管理ユニットであることを特徴とする、請求項１４に記載のフォールト・トレラント・コンピュータ・システム。

【請求項１６】　前記第１記録機構が、前記第１記録機構を活動化するための同期はずれ信号を受領するように接続されたイネーブル入力を有することを特徴とする、請求項１５に記載のフォールト・トレラント・コンピュータ・システム。

【請求項１７】　前記第２記録機構が、同期はずれイベントに続いて前記第１記録機構を活動化する時間をカバーするのに十分な数まで、最近のメモリ更新イベントの記録を維持することを特徴とする、請求項１４に記載のフォールト・トレラント・コンピュータ・システム。

【請求項１８】　前記第２記録機構が先入れ先出しバッファを含むことを特徴とする、請求項１７に記載のフォールト・トレラント・コンピュータ・システム。

【請求項19】　前記第1記録機構が前記先入れ先出しバッファの出力に接続されていることを特徴とする、請求項18に記載のフォールト・トレラント・コンピュータ・システム。

【請求項20】　前記先入れ先出しバッファが、更新アドレスを所定数まで格納し、アドレス・デコーダが、前記先入れ先出しバッファの前記出力に接続されて、前記先入れ先出しバッファから出力されるメモリ更新アドレスを表すページ信号を生成し、前記アドレス・デコーダが、前記同期はずれ信号に反応して、前記ページ信号を前記第1記録機構に移すことを特徴とする、請求項19に記載のフォールト・トレラント・コンピュータ・システム。

【請求項21】　前記第2記録機構が論理解析器を含むことを特徴とする、請求項14に記載のフォールト・トレラント・コンピュータ・システム。

【請求項22】　前記第2記録機構が、同期はずれイベントに続いて前記第1記録機構を活動化する時間をカバーするのに十分な数まで、最近のメモリ更新イベントの記録を維持し、前記第2記録手段の動作が、前記同期はずれ信号に応答して阻止されることを特徴とする、請求項14に記載のフォールト・トレラント・コンピュータ・システム。

【請求項23】　前記第1記録機構がソフトウェアが生成する更新イベントのリストを有することを特徴とする、請求項22に記載のフォールト・トレラント・コンピュータ・システム。

【請求項24】　前記第2記録機構が、主メモリ中で維持されるアドレス変換バッファおよびメモリ・アクセス表を含むことを特徴とする、請求項14に記載のフォールト・トレラント・コンピュータ・システム。

【請求項25】　前記メモリ再統合機構が、前記第1および第2記録機構で識別されたメモリ・ページを再統合するように動作可能であることを特徴とする、請求項14に記載のフォールト・トレラント・コンピュータ・システム。

【請求項26】　ロックステップ方式で動作する3つの同期処理セットを含み、前記同期はずれ検出器が同期はずれ処理セットを多数決により決定することを特徴とする、請求項14に記載のフォールト・トレラント・コンピュータ・システム。

【請求項27】　前記同期はずれ検出器が、残りの2つの処理セットのうち1つを選択し、同期はずれ処理セットおよび残りの処理セットへの入力を供給して前記同期はずれ処理セットおよび残りの処理セットのアイドルを引き起こし、メモリ書込みイベントのソフトウェア・ログを維持しながら、前記同期はずれ処理セットおよび残りの処理セットのうち1つを再統合し、続いて前記ソフトウェア・ログを使用して、前記同期はずれ処理セットおよび残りの処理セットのうち前記のもう1つを再統合するように配列されることを特徴とする、請求項26に記載のフォールト・トレラント・コンピュータ・システム。

【請求項28】　個々にプロセッサおよび内部メモリを含み、ロックステップ方式で動作する複数の同期処理セットと、障害イベントを検出し障害信号を生成する障害検出器とを含むフォールト・トレラント・コンピュータ・システムの処理セットを再統合する方法であって、限られた期間にわたってメモリ更新イベントの一時記録を維持する段階と、前記障害信号に応答して、前記障害状態に続いてメモリ更新イベントの後続記録を活動化する段階と、障害の発生したプロセッサ中で、少なくとも前記一時メモリ記録および後続メモリ記録で識別される部分のメモリに対して、メモリ再統合を実施する段階とを含む、フォールト・トレラント・コンピュータ・システムの処理セットを再統合する方法。

【請求項29】　前記障害イベントが同期はずれイベントであることを特徴とする、請求項28に記載の方法。

【請求項30】　後続記録がメモリ管理ユニットの記憶装置に格納され、前記記憶装置中のページ・エントリがメモリの各ページに対して与えられ、コードが、前記第1記録機構が活動化されてページが書込みされるたびに、そのページのエントリに書込みされることを特徴とする、請求項28に記載の方法。

【請求項31】　前記後続記録を活動化する時間をカバーするのに十分な数まで、最近のメモリ更新イベントの記録を維持する段階を含む、請求項28に記載の方法。

【請求項32】　前記一時記録が先入れ先出しバッファに格納されることを特徴とする、請求項31に記載の方法。

【請求項33】　後続記録のための記録機構を、前記先入れ先出しバッファの出力に接続する段階を含む、請求項32に記載の方法。

【請求項34】　前記先入れ先出しバッファに所定数まで更新アドレスを格納する段階と、前記先入れ先出しバッファの出力をアドレス・デコーダに供給して、前記先入れ先出しバッファから出力されたメモリ更新アドレスを表すページ信号を生成する段階と、前記障害信号が活動状態になったときに、前記ページ信号を前記後続記録の一部として記録する段階とを含む、請求項33に記載の方法。

【請求項35】　前記一時記録が論理解析器に格納されることを特徴とする、請求項31に記載の方法。

【請求項36】　前記障害信号に応答して阻止される最近のメモリ更新イベントの前記一時記録を、障害に続いて前記第1記録機構を活動化する時間をカバーするのに十分な数まで維持する段階を含む、請求項31に記載の方法。

【請求項37】　ソフトウェアによって更新イベントのリストを生成する段階を含む、請求項36に記載の方

法。

【請求項38】　主メモリによって維持される表検索バッファおよびメモリ・アクセス表によって、前記一時記録を形成する段階を含む、請求項31に記載の方法。

【請求項39】　前記一時記録および後続記録中で識別されたメモリ・ページを再統合する段階を含む、請求項31に記載の方法。

【請求項40】　ロックステップ方式で動作する3つの同期処理セットを含み、同期はずれ検出器が同期はずれ処理セットを多数決によって決定することを特徴とする、請求項31に記載の方法。

【請求項41】　同期はずれ処理セットの識別に応答して、残りの2つの処理セットのうち1つを選択する段階と、同期はずれ処理セットおよび残りの処理セットへの割込みを供給して前記の同期はずれ処理セットおよび残りの処理セットのアイドルを引き起こす段階と、メモリ書込みイベントのソフトウェア・ログを維持しながら、前記の同期はずれ処理セットおよび残りの処理セットのうち1つを再統合する段階と、続いて前記ソフトウェア・ログを使用して、前記の同期はずれ処理セットおよび残りの処理セットのうち前記のもう1つを再統合する段階とを含む、請求項31に記載の方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、一般に、同様に動作する複数のサブシステムを使用する、ロックステップ（lockstep）方式のフォールト・トレラント・コンピュータなどのフォールト・トレラント・コンピュータ・システムに関する。

【0002】

【従来の技術】このようなロックステップ方式のフォールト・トレラント・コンピュータ・システムでは、コンピュータ内で各サブシステムの出力を比較し、これらの出力が異なる場合には、何らかの例外的な修理処置を行う。

【0003】添付の図面の図1は、典型的なシステムの一例を示す概観図であり、3つの同じ処理（ＣＰＵ）セット10、11、12が、共通のクロック16の下で同期（ｓｙｎｃ）して動作する。1つの処理セットは、処理エンジン、例えば中央処理装置（ＣＰＵ）や内部状態記憶装置を含む1つのサブシステムを表す。添付の図面の図2は、このような処理セットの概略図である。この図は、処理エンジン20、内部状態記憶装置（メモリ）22、および内部バス23を示している。処理セットは、コンピュータ・システムのその他のエレメントを含むことがあるが、通常は、入出力インタフェースは含まないことになる。外部接続、例えば内部バス13からの接続13、外部クロック16用の入力15、およびハードウェア割込み入力14も設けられている。

【0004】図1に示すように、3つの処理セット10、11、12の出力は、処理セット10、11、12の動作を監視する、障害検出ユニット（ボータ）17に供給される。処理セット10、11、12が正常に動作している場合は、これらは同じ出力を生成し、ボータ17に送る。したがって、出力が合致している場合は、ボータ17は、処理セット10、11、12から入出力（Ｉ／Ｏ）サブシステム18へコマンドを通過させ、動作させる。しかし、各処理セットからの出力が異なる場合は、これは何かが不適当であることを示し、ボータは、Ｉ／Ｏ動作に影響を与える前に、何らかの補正処置を行う。

【0005】通常は、補正処置には、ボータが適当なライン14を介して障害を示す処理セットに信号を供給し、障害のある処理セットを照らす「自己変更」光（図示せず）を生じさせる処置が含まれる。欠陥のある処理セットはスイッチ・オフになり、続いて操作員は、これを正常に機能するユニットと交換しなければならない。図示した例では、1つの処理セットが一時的または永続的な障害をもたない、またはこれを発生させる場合には、1対2の投票結果が現れることになるので、通常は、欠陥のある処理セットを多数決によって容易に識別することができる。

【0006】しかし、本発明はこのようなシステムに限定されるものではなく、障害のある処理セットを識別するために拡張診断動作が必要となるシステムにも適用可能である。このシステムは、単一のボータを備える必要がなく、入出力コマンドのみを投票で決める必要がない。本発明は、一般に、ロックステップ方式で動作する冗長部品を備えた同期システムに適用可能である。

【0007】ロックステップ方式のシステムは、フォールト・トレラント処理コアを構成する処理セットの全体の同期に依存する。したがって、処理セットは同様に動作するハードウェアを必要とし、さらに、処理セットのデータの内部記憶状態もまた、同様であることが必要とされる。新しい処理セットを動作中のシステムに統合する過程の一部には、動作中のシステムの主メモリの内容を、新しい処理セットにコピーする段階が含まれる。主メモリは、非常に大きい、例えばギガバイトの単位になることも可能なので、この過程は計算に関してかなり長い時間がかかる可能性がある。

【0008】ロックステップ方式のコンピュータ・システムは、様々な原因から同期はずれになる可能性がある。主な原因は、永久的な単一の処理セットの故障である。このような故障からの回復は、故障したユニットの除去、機能するユニットとの交換、および機能するユニットの復旧を含む。新しい処理セットが、動作中の処理セットのメモリの内容について何も知っておらず、動作中のシステムからの全ての主メモリを、新しい処理セットにコピーしなければならないことは明らかである。

【0009】別法として、障害度の低い同期はずれイベ

ントは、動作中のコンピュータ・システムがしばしば自動的に診断することができ、同期はずれ処理セットを交換することなく、これを自動的に再統合することができる。例えば、宇宙線イベントによって引き起こされる可能性のある動的メモリ中のソフト・データ・エラーは、小規模な動作の混乱を引き起こす可能性があるが、これは自動的に修復することができる。しかし、これは、依然として同期はずれ処理セットのメモリ状態の再統合を必要とし、それは動作中のシステムの主メモリの内容を、同期はずれ処理セットにコピーすることである。したがって、主メモリが非常に大きいことがあり得るので、これは計算に関して、依然としてかなり長い時間がかかる可能性がある。

【００１０】

【発明が解決しようとする課題】本発明は、従来技術の問題を回避する、小規模な同期はずれイベントから回復する自動的かつ迅速な方法を提供しようとするものである。

【００１１】

【課題を解決するための手段】本発明の一態様によれば、フォールト・トレラント・コンピュータ・システム用のメモリ管理システムが提供される。このメモリ管理システムは、活動化されてメモリ更新イベントを記録することができる第１記録機構と、少なくとも限られた数のメモリ更新イベントを記録するだけの容量を備えた第２記録機構と、障害イベントの場合に第１記録機構を活動化するための障害信号用の障害入力と、少なくとも第１および第２記録機構で識別された部分のメモリを再統合するメモリ再統合機構とを含む。

【００１２】本発明の実施態様は、ロックステップ方式のシステムの処理セット間の小規模な同期はずれイベントの後、同期はずれ処理セットのメモリ内容の大部分が、最初のうちは動作中のシステムのメモリ内容と同じであることを利用する。同期はずれ処理セットまたは動作中のシステムのどちらでも、そのメモリ・システム内の比較的少数のロケーションのみが修正されていることになる。しかし、動作中のシステムが通常処理ロードの動作及び実行を継続するにつれて不一致（divergence）が、時間とともに拡大する。本発明の実施形態により、不一致を追跡して補償し、さらに、同期はずれイベントの前後、および第１記録機構が活動化される以前の任意のメモリ更新イベントが捕捉することが可能になる。

【００１３】メモリ更新（書込み）の記録は、アクセスされた個々のアドレスの記録に基づくのではなく、更新された（書込みされた）メモリ・セグメント（ページ）に基づいていることが好ましい。換言すれば、第１および／または第２記録機構は、更新された（書込みされた）セグメント（すなわちページ）を記録することが好ましい。これは、セグメント（ページ）あたり１ビットを備えたセグメント（すなわちページ）・メモリを使用して、書き込まれたセグメント（ページ）を識別することによって、効果的に行うことができる。

【００１４】本発明の別の態様によれば、個々に内部メモリを備えたプロセッサを含み、ロックステップ方式で動作する複数の同期処理セットと、同期はずれイベントを検出し、同期はずれ信号を生成する同期はずれ検出器とを含むフォールト・トレラント・コンピュータ・システムが提供される。個々の処理セットはまた、活動化されてメモリ書込みイベントを記録することができる第１記録機構と、少なくとも限られた数のメモリ書込みイベントを記録するだけの容量を備えた第２記録機構と、同期はずれイベントの場合に、同期はずれ信号を受領して第１記録機構を活動化する障害入力と、少なくとも第１および第２記録機構で識別された部分のメモリを、同期はずれ処理セット中で再統合するメモリ再統合機構とを含む。

【００１５】本発明の別の態様によれば、障害に続いて、フォールト・トレラント・コンピュータ・システムの処理セットを再統合する方法が提供され、このフォールト・トレラント・コンピュータ・システムは、個々にプロセッサおよび内部メモリを含み、ロックステップ方式で動作する複数の同期処理セットと、障害イベントを検出し障害信号を生成する障害検出器とを含む。この方法は、限られた期間にわたってメモリ更新イベントの一時記録を維持する段階と、障害に応答して、障害状態に続いてメモリ更新イベントの後続記録を活動化する段階と、障害の発生した処理セット中で、少なくとも一時メモリ記録および後続メモリ記録で識別された部分のメモリに対して、メモリ再統合を実施する段階とを含む。

【００１６】本発明の一実施形態では、同期はずれイベントの後の、少なくとも選択された主メモリへのメモリ・アクセス・イベント（メモリ書込みイベント）の記録は保存されるので、同期はずれ処理セットを再統合するには、修正されたメモリ・ロケーションのみをコピーすればよい。

【００１７】本発明の一実施態様では、第１記録機構は、複数のメモリ・ページのそれぞれに対するエントリと、第１記録機構が活動化されてページが書込みされるたびに、そのページのエントリに書込みされるコードとを備えたＲＡＭを含むメモリ管理ユニットである。

【００１８】第１記録機構は、障害（同期はずれ）信号を受領するように接続されたイネーブル入力を有することが好ましい。

【００１９】第１記録機構は、処理セット中の全ページ数までの任意数の多数の書込みページを記録することができる。

【００２０】第２記録機構は、障害イベントに続いて第１記録機構を活動化する時間をカバーするのに十分な数まで、最近のメモリ更新イベントのローリング記録を維持することが好ましい。

【0021】第2記録機構は、先入れ先出しバッファを含むことができ、1つの実施形態の第1記録機構は、この先入れ先出しバッファの出力に接続されている。この構成では、先入れ先出しバッファは更新アドレスを所定数まで格納し、アドレス・デコーダは、先入れ先出しバッファの出力に接続されて、この先入れ先出しバッファから出力されるメモリ更新アドレスを表すページ信号を生成することができ、アドレス・デコーダは、同期はずれ信号に反応して、このページ信号を第1記録機構に移す。

【0022】別法として、第2記録機構は、論理解析器を含むことができる。これにより、フォールト・トレラント・コンピュータが、通常は障害の解析用に論理解析器を含むので、実施コストを削減することができる。

【0023】第2記録機構の出力が第1記録機構への入力にならない場合は、第2記録手段の動作が、同期はずれ信号に応答して阻止されることが好ましい。

【0024】第1記録機構は、ソフトウェアによって生成された表を含むことができ、この表では、ページが書込みされるたびに、メモリのそのページに対応する記録がコードでマークされる。この記録は、プロセッサの変換索引バッファ中のエントリを更新するソフトウェアによって維持することができる。第2記録機構は、最近TLBから出たページのリストを備えたTLBの内容であることがある。

【0025】同期はずれ入力に応答して、ソフトウェアは、最近書込みされたページを求めてTLBおよびリストを探索することができ、それらを第1記録機構中で書込み済みとしてマークし、続いて、処理ユニットが再統合されるまで、第1記録機構を維持し続けることがある。

【0026】メモリ再統合機構は、第1および第2記録機構で識別されたメモリ・ページを再統合するように動作可能であることが好ましい。

【0027】本発明は、同期はずれ検出器が同期はずれ処理セットを多数決で決定するような、ロックステップ方式で動作する3つの同期処理セットを含むコンピュータ・システムに適用可能である。

【0028】この場合は、同期はずれ処理セットの再統合は、同期はずれ処理セットの識別に応答して、残りの2つの処理セットのうち1つを選択し、同期はずれ処理セットおよび残りの処理セットをアイドルにさせるために同期はずれ処理セットおよび残りの処理セットへの割込みを供給し、メモリ書込みイベントのソフトウェア・ログを維持しながら、同期はずれ処理セットおよび残りの処理セットのうち1つを再統合し、かつその後ソフトウェア・ログを使用して、同期はずれ処理セットおよび残りの処理セットのうちもう1つを再統合することによって達成することができる。

【0029】前述の発明は、ロックステップ方式のフォールト・トレラント・コンピュータの処理セットの再統合時間を数十分から何分の1秒かに削減することができる。再統合期間中、コンピュータは、動作中の処理セットの後続の障害に対して無防備である。したがって、再統合時間の削減は、コンピュータの全体的な有用性に関してかなり有益である。

【0030】

【発明の実施の形態】本発明の実施形態について、同様の参照符号が同様の特徴に関係する添付の図面に関連して、以下に述べる。

【0031】図3は、機能に関して本発明の一例のエレメントを表す、概略的なブロック・ダイアグラムである。図3は、一般に、例えば図1に示すシステムなどのフォールト・トレラント・コンピュータ・システム用の処理セット10／11／12のうち1つを表す図である。

【0032】図3では、処理エンジン（例えば中央処理装置（CPU））20および内部状態記憶装置（メモリ）22は、内部バス23で接続される。外部接続、例えば内部バス23からの接続13、外部クロック用の入力15、およびハードウェア割込み入力14も設けられる。

【0033】図3には、活動化されてメモリ更新イベントを記録することができる第1記録機構25、少なくとも限られた数のメモリ更新イベントを記録するだけの容量を備えた第2記録機構26、および少なくとも第1および第2記録機構で識別された部分のメモリを再統合するメモリ再統合機構もまた、概略的に示す。図3に示すように、機構25、26、27のそれぞれは、内部バス23に接続されて示されている。これは、第1記録機構25および第2記録機構26が、メモリ・アクセス・イベントを監視して、メモリ更新（メモリ書込み）が起こった時間および位置を識別する必要があるためである。再統合機構27もまた、第1および第2記録機構にアクセスして、同期はずれ処理セットおよび動作中の処理セットのメモリ中で、メモリ書込みが起こった位置を判別し、続いて対応するメモリ部分を、動作中の処理セットのメモリから同期はずれ処理セットのメモリへコピーする必要がある。ただし、機構25、26、および27は、以下の記述で説明するように、様々な方法で実施することができる。様々な実施形態には、ハードウェアおよびソフトウェアの様々な組合せが含まれ、様々なエレメントの相互接続は、通常は図3に図示したものとは異なることになる。例えば、再統合機構は通常はソフトウェアで実施されることになり、ポータ17と連結している、および／またはポータ17の一部をなす、制御コンピュータ中で実施されることがある。また、第1記録機構25は、バス23に直接接続されずに、第2記録機構26を介して接続されることがある。第1および第2記録機構もまた、以下に記述するように、程度の違いはあ

るがソフトウェアで実施されることがある。

【００３４】コンピュータ・システムは、通常は主メモリのトラックを保持し、その使用を制御するために、メモリ管理ハードウェアを含む。メモリを指定サイズのページに分割し、個々のページへのアクセス制御の小さな記録を保存することもまた、一般的である。ハードウェア機構もまた、そのページが修正されていることを示すビットを備えたページについての記録を更新するために存在する。このビットは、ページについてのいわゆる「ダーティ」ビットである。メモリのページは、最初の状態から変化させるための書込みがされていない場合は「クリーン」と呼ばれ、そのような書込みがされた後は「ダーティ」と呼ばれる。ソフトウェアは、メモリ管理ユニットの記録中のページに対するダーティ・ビットを一掃することによって、そのページが「クリーン」とマークされるようにすることができる。その後にハードウェアがそのビットを１に設定し、そのページが書込みされたことを示すことになる。通常の動作では、メモリ管理ユニットは、コンピュータ・メモリの多くのページを、そのほとんどの時間でダーティであると見なすことになる。したがって、従来の方法で動作する従来のメモリ管理ユニットが、ロックステップ方式のフォールト・トレラント・コンピュータ・システムの個々の処理セットに備えられている場合は、同期はずれイベントが起こったときに、多くのページがダーティとマークされることになる可能性が高い。

【００３５】従来通りに構成されたコンピュータ処理セットのメモリ管理ユニットは、通常は動作中のオペレーティング・システムの制御下にあるので、本発明の最初の実施形態では、単に同期はずれイベントの後で処理セットを再統合するソフトウェアを使用するために、追加のメモリ管理ユニットを備える。

【００３６】図４は、メモリのどのページがダーティであり、どのページがクリーンであるかという情報のみを含むようにカスタマイズされている、従来のメモリ管理ユニット４０を示す図である。以下の記述では、このタイプのメモリ管理ユニットを「ダーティＲＡＭ」と呼ぶ。ソフトウェア４２はダーティＲＡＭ記憶装置４６にアクセスして、どのページがダーティであるかを検査することができ、そこに直接書込みを行ってページの状態をダーティまたはクリーンに変更することができる。さらに、ハードウェア４４は、バス２３を介して書込みをされた主メモリの任意ページに対する記録の状態を、自動的にダーティに変更する。この実施形態では、ダーティＲＡＭ記憶装置４６のただ１つのビットのみが、主メモリの個々のページ全体に対して使用される。ダーティＲＡＭが監視する「ページ」サイズは、システム中のその他のメモリ管理ユニットで使用するサイズと同じである必要はないが、全てのページが同じサイズであることは、しばしば便利かつ効率的である。コンピュータはページ単位で働く傾向があり、ページの一部に書込みアクセスすることは、しばしば同じページ内のその他のロケーションにも書込みをすることになることを意味する。しかし、図３に示す従来のメモリ管理ユニットでは、前述のように通常はほとんどのページがダーティであるので、このタスクを制御して実施するためには、本来不十分であることになる。

【００３７】図５は、本発明の最初の実施形態に関するダーティＲＡＭ５０の、概略的なブロック・ダイアグラムである。図５では、ダーティＲＡＭ５０は独立したイネーブル入力５８を備え、それによってハードウェア５４は、処理セットが同期はずれになった後でダーティＲＡＭ記憶装置５６中のダーティ・ページのロギングを単に開始する。イネーブル入力上の信号は、ボータ１７で同期はずれイベントが検出されるのに応答してアサートされる。

【００３８】ダーティＲＡＭイネーブル入力５８によって、ダーティＲＡＭシステムの動作が可能になる。通常の動作では、処理セットが同期して動作している場合は、ダーティＲＡＭイネーブル入力はアサートされず、ダーティＲＡＭ５０は、ソフトウェア５２によって、全てのページに「クリーン」状況が与えられるように設定される。

【００３９】同期はずれイベントが起こったとき、イネーブル入力５８はアサートされるようになる。イネーブル入力がアサートされる間（すなわち処理セットが同期はずれである間）に、ダーティＲＡＭは、書込みされる主メモリのページをロギングする。書込みされるページは、動作中の処理セットおよび同期はずれの処理セット上のものとは異なる可能性があるページであることになる。図５に示すイネーブル入力を備えたダーティＲＡＭは、個々の処理セットに備えられ、そこで各システム・バス２３に接続される。処理セットが同期して動作している間、個々のダーティＲＡＭは、ソフトウェア５２によってクリーン状態に保持される。処理セットが同期はずれ状態で動作していることが検出されると、ダーティＲＡＭのロギングが可能になる。この実施形態では、ハードウェア・イネーブル信号５８は、処理セット１０、１１、１２が同期はずれであることを検出すると同時に、同期はずれ検出ハードウェア（すなわちボータ１７）中で生成される。換言すれば、処理セットからの出力中の差分を検出すると直ちに、ボータは、個々の処理セットに供給されてアサートされた同期はずれ信号になる信号を生成する。アサートされた後は、同期はずれ信号は、処理セットが復旧するまで否定されない。その他の実施形態では、ソフトウェアによってイネーブル信号を生成することができる。

【００４０】同期はずれイベントの後、ソフトウェアおよび／またはハードウェア機構は、フォールト・トレラント・コンピュータ・システムを再構成するように動作

する。このシステムは少なくとも1つの処理セットで通常動作の動作を続行する。同期はずれ処理セットを含めた少なくとも1つの処理セットは、動作しなくなる。その結果、この同期はずれ処理セットは通常動作の動作を停止し、動作中のシステムに再統合されるのを待つ。動作中の処理セットおよび同期はずれ処理セットにメモリ書込みを行うと、動作中の処理セットおよび同期はずれ処理セットの主メモリの内容に不一致が発生する。

【0041】動作中のシステム上のソフトウェアは、同期はずれ処理セットの再統合に取りかかると、同期はずれイベント以降にダーティになったメモリ・ページを発見するために、動作中のシステム上のダーティRAMにアクセスする。これはまた、同期はずれ処理セット中のダーティRAMにもアクセスする。このダーティRAMにより、不一致が始まってから、どのページが同期はずれプロセッサによって修正されているかが分かる。ダーティRAMのうちのいずれかで、メモリのあるページがダーティであると挙げられた場合、動作中の処理セットおよび同期はずれ処理セット上で、再統合ソフトウェアでそのページをコピーし、同期はずれ処理セットを同期状態に戻さなければならない。任意のダーティRAM中でメモリのページがダーティであるとしてマークされていない場合には、同期はずれ処理セット上にあってもそのページは依然として妥当であることになるので、無視することができる。

【0042】本発明の代替の実施形態では、処理セットは、常にダーティ・ページをロギングするように動作している、イネーブル・ピンを備えないダーティRAM記憶を備える場合、ソフトウェアは、同期はずれイベントに基づくハードウェア信号によって活動化され、ダーティRAMを一掃することができる。このソフトウェアは、クリーニング過程中にそれ自体が汚したページを慎重に記録しなければならない。

【0043】さらに別の代替の実施形態では、通常のメモリ管理ユニットもまた、ダーティ・ページ情報を収集するために使用することができる。この代替の実施形態では、ソフトウェアは、主メモリの全てのページが書込み禁止になるように、同期はずれイベント時にページ表を修正するように構成される。このことは、メモリへの書込みサイクルが、プロセッサのバス・エラー例外になることを意味する。続いてプロセッサは、最初に各バス・エラーに作用して、ソフトウェアが維持するダーティ・ページのリストに書込みページを追加し、その後そのページに対する書込み禁止を解除して、そこへの将来の書込みが正常に完了するようにする。このことには、クリーン・ページを探索して偶発的なダーティ・ページを探すことなく、ダーティ・ページの単一のリストのみを再統合ソフトウェアで検査すればよいという利点がある。

【0044】ただし、追加のソフトウェアを備えた従来のメモリ管理ユニットを使用して、ダーティ・ページ情報を収集するのではなく、独立した「ダーティ・メモリ」を備えることが望ましいことに留意されたい。これは、従来のメモリ管理ユニットには2つの欠点があるためである。1つ目は、従来のコンピュータ・オペレーティング・システム・ソフトウェアは、それ自体の目的のためにメモリ管理ユニットを使用することである。2つ目は、従来のメモリ管理ユニットは、単一のプロセッサでしか機能せず、多重プロセッサの動作、またはI／O装置による直接メモリ・アクセスをカバーすることができないことである。

【0045】どのような方法を利用して、ダーティ・メモリ・ページ上のデータを収集しても、同期はずれイベントの付近に問題がある可能性は高い。同期はずれイベントの検出とダーティRAMのデータ収集の間に、時間がいくらか経過する必要があり、この期間中に少数のダーティ・ページは記録されないままであることがある。厳密にどれだけの数のページが記録されていないかは、ダーティRAMの実施態様によって決まるが、単一のページを記録していないだけでも、同期はずれイベントの後で、全てではなくいくつかの主メモリのみをコピーする方式を役に立たなくするには十分である。

【0046】したがって、本発明の実施形態では、同期はずれイベントの前後でメモリ書込みイベントを記録するための機構もまた備える。

【0047】ダーティ・ページの記録を提供する前述の機構は、同期はずれイベント時から動作をし、同期はずれイベントに続いて、必要な全てのページを記録することができる。しかし、これを補って独立した一時の記録にすることが、同期はずれイベントの付近にダーティにされたページに対して必要とされる。この独立した記録は、限られた時間にわたって、好ましくはローリング式に、書込みイベントを考慮しなければならない。この独立した記録は、同期はずれイベント自体から、前述の機構が記録を開始することができる時間までの間に発生する可能性がある書込みイベントを収容するのに十分な容量を備える必要がある。この独立した記録を、以下の記述では第2ダーティ・ページ記録と呼び、前述の「ダーティRAM」と区別する。

【0048】障害によって同期はずれが送出される時を予言することができないので、第2ダーティ・ページ記録は、（少なくとも同期はずれイベントが発生するまで）継続して動作している必要がある。同期はずれイベントの直前または直後にダーティにされ、ダーティRAMが正確に収集することができないページを記憶することは、第2ダーティ・ページ記録の仕事である。第2ダーティ・ページ記録はまた、限られた時間メモリも備える必要がある。無期限のはるか過去にダーティにされたページを記憶すると仮定すると、最終的には、全てのメモリ・ページをダーティであるとしてリストすることに

なる。主ダーティ・ページ記憶ではとらえることができない、ダーティにされたページが確実に含まれるのに十分なだけ過去に逆戻って、同期はずれイベントを記憶すればよい。

【００４９】以下に述べるいくつかの実施形態では、第２ダーティ・ページ記録がダーティＲＡＭと平行して動作する場合には、第２ページ記録は、同期はずれイベントの時点で、またはその直後に停止する。これらの実施形態では、動作状態のままであると仮定すると、時間が限られているというこの記録の性質のために、最終的に同期はずれイベントについての重要な情報が上書きされる、または失われる可能性がある。これは、ダーティＲＡＭのイネーブル信号のアサートに応答して第２ページ記録の動作を阻止することによって、都合よく行うことができる。

【００５０】第２ダーティ・ページ記録が停止した後は、ソフトウェアまたはハードウェアのどちらでも、これを検査すること、およびそこにリストされたダーティ・ページを、主ダーティＲＡＭまたは独立したリストに追加し、再統合ソフトウェアでコピーすることができる。同期はずれ処理セットおよび動作中の処理セットは両方とも、第２ダーティ・ページ記録を有する。ソフトウェアはそれらの記録を検査および比較することができ、必要ならば、どのページが実際に処理セットによって同期してダーティにされたかを導き出すことができる。これにより、コピーするページ数が減少することになる。

【００５１】１つの実施形態では、論理解析器を使用して、第２ダーティ・ページ記録に関する情報を収集する。図６は、処理セットのバス２３を観測する論理解析器６０を示す図である。トリガ機構６６、クロック修飾子６２、およびアドレス生成プログラム６４を備えた論理解析器６０は、各処理セットに備えられる。論理解析器は、通常は動作状態である。解析器６０がトリガされると、処理セットの同期はずれ信号のアサートによってデータ収集の停止が引き起こされ、同じ信号が主ダーティＲＡＭにデータ収集を開始させる。論理解析器は、最終的に動作を停止し、同期はずれイベントの前後両方のコンピュータ・バスの動作の記録を保存する。同期はずれイベントの後で、同期はずれ処理セットおよび動作中の処理セットからの論理解析器のトレースを解析することによって、記憶されたどのトランザクションが不一致書込みサイクルであるかを導き出すことができる。その後、ソフトウェアが維持する第２ダーティ・ページ記録中で書込みされた１組のページに、関連したページを追加することができる。論理解析器は、書込みされたページを判別することができるように、少なくとも各バス・サイクルについてのアドレスおよび制御情報を記憶する必要がある。論理解析器の出力の解析は、ソフトウェア・ルーチンによって容易に行うことができる。

【００５２】第２ダーティ・ページ記録に対して論理解析器を使用することの利点は、ロックステップ方式のフォールト・トレラント・コンピュータには、通常は、障害診断のために論理解析器が組み込まれており、同期はずれイベントと同時にトリガされることになる点である。この場合は、前述のように、論理解析器の出力を制御し、解析するためのソフトウェアを備えるだけでよい。

【００５３】代替の実施形態として、書込みバッファが、第２ダーティ・ページ記録用の記憶装置を提供することができる。図７は、内部バス２３を介した主メモリへの書込み用の、短期バッファとして使用される先入れ先出しメモリ７０を示す図である。通常の同期状態の動作では、主メモリへの書込みは書込みデコード論理７１でデコードされ、各書込みのページ番号がＦＩＦＯ７０に書込みされる。同期はずれが起こると、ハードウェア同期はずれ検出信号５８は、ＦＩＦＯ７０への後続の書込みを阻止する。その後に、ソフトウェアは、ＦＩＦＯ７０の内容を検査して、ダーティ・ページのリストにページを追加することができる。第２ダーティ・ページ記録用の書込みバッファの利点は、ソフトウェアおよびハードウェアのどちらについても、論理解析器より単純である点である。

【００５４】さらに別の代替の実施形態では、書込みバッファをダーティＲＡＭと直列に配列することができる。

【００５５】図８は、書込みバッファがダーティＲＡＭと直列に配列される、第１および第２記録機構の組合せの第１例を示す図である。図８の配列は、図４および図７の配列の組合せに基づいている。この場合は、ＦＩＦＯバッファ８０は、処理セットのメモリへの書込みイベントを記録するために継続して動作し、この書込みイベントは、書込みデコード論理８１によって継続してデコードされ、ＦＩＦＯ８０内の記憶装置にページ・アドレスを供給する。この場合では、後述のように、書込みデコード論理が阻止入力を受領する必要はない。ＦＩＦＯ８０に供給されるページ・アドレスは、遅延の後で、ＦＩＦＯ８０の出力に現れる。しかし、これらのページ・アドレスは、ゲート８４がライン５８上の同期はずれ信号によってイネーブルになるまで、このゲートによってダーティＲＡＭ記憶装置８６に移ることを妨げられる。この同期はずれ信号は、効果的にダーティＲＡＭにイネーブル信号を与え、その後この信号によって、ＦＩＦＯ８０からのページ・アドレスをダーティＲＡＭ記憶装置８６に供給することが可能になり、それにより、適当なページ・ビットを設定することができる。ソフトウェア８２を使用して、同期はずれイベントの前の任意の時点でダーティＲＡＭ記憶装置８６を一掃し、同期はずれ信号が供給されたときに、それが「クリーン」であるようにすることができる。

【００５６】この実施形態では、ＦＩＦＯ８０のサイズおよび書込みイベントの頻度によって決まる時間の後で、ＦＩＦＯバッファ８０の内容がダーティＲＡＭに自動的に記憶されるので、ＦＩＦＯバッファ８０を機能停止にする必要はない。この実施形態では、再統合機構は、ダーティＲＡＭ８６およびＦＩＦＯ８０を考慮に入れることが好ましい。

【００５７】図９は、書込みバッファがダーティＲＡＭと直列に配列された、第１および第２記録機構の組合せの第２例を示す図である。この例では、ＦＩＦＯバッファ９０は、処理セットのメモリへの書込みイベントを記録するために、継続的に動作している。ＦＩＦＯバッファ９０からの出力は、同期はずれイネーブル信号５８に応答してのみイネーブルになる、アドレス・デコーダ９１に供給される。アドレス・デコーダ９１がイネーブルでないとき、ＦＩＦＯバッファの出力は効果的に廃棄される。アドレス・デコーダがイネーブルであるときのみ、ダーティ・ページ・ビットはアドレス・デコーダ９１から出力され、ダーティＲＡＭ記憶装置９６中の適当なページ・ロケーションに記憶される。

【００５８】任意選択的に、同期はずれイネーブル信号はまた、ダーティＲＡＭ自体にも供給されるが、アドレス・デコーダに同期はずれ信号を供給すれば、効果的にダーティＲＡＭにイネーブル信号が提供されることは理解されるであろう。図８の例と同様に、この実施形態では、ＦＩＦＯ９０のサイズおよび書込みイベントの頻度によって決まる時間の後で、ＦＩＦＯバッファ９０の内容がダーティＲＡＭに自動的に記憶されるので、ＦＩＦＯバッファ９０を機能停止にする必要はない。この実施形態では、再統合機構は、ダーティＲＡＭ９６およびＦＩＦＯ９０を考慮に入れることが好ましい。

【００５９】さらに、ソフトウェアが実施する実施形態では、アドレス変換バッファ（ＴＬＢ）および関連するＴＬＢミス・ルーチン、加えて主メモリで作成されるダーティ・ページ記憶を使用する。ＴＬＢは、ページ化されたメモリを使用するほとんどのコンピュータ・アドレッシング方式の標準的な部品である。いくつかのコンピュータは、固定ハードディスクではなく、ソフトウェアＴＬＢミス・ルーチンで、ＴＬＢエントリを維持する。この実施形態では、障害イベントに続いて、ソフトウェアは、書込み可能なページを現在指定しているＴＬＢエントリに注目し、それらのエントリをソフト・ダーティ・ページ記憶に追加することができる。ソフトウェアはまた、これに、最近ＴＬＢから出た書込み可能なページのリストを転送することもできる。このリストは、障害イベントの前は通常の環境でミス・ルーチンによって維持され、障害イベントの付近に書込みされて直ちにＴＬＢから出た可能性のあるページを示す。これに続いて、再統合の進行中に、ＴＬＢミス・ルーチン中のソフトウェアは、書込みされた各ページを、ソフト・ダーティ・ページ記憶に追加する。このようにして、ダーティ・ページの記録を作成することができる。

【００６０】前述の手法は、モジュール三重化（ＴＭＲ）システムに適用することができる。いくつかのロックステップ方式のＴＭＲシステムは、同期はずれイベントが起こったときに、単一の処理セットで動作する状態に切り替わる。これは、回復するために、独立した２つの再統合段階を必要とする。

【００６１】ＴＭＲシステムが、処理セット１０、１１、および１２とともに同期して動作している例について記述する。この例では、再統合は、ボータ１７の一部をなす制御コンピュータの制御下で、ソフトウェアによって実施される。

【００６２】この例では、処理セット１２に、このシステムを同期はずれにするＲＡＭソフト・エラーがあるものと想定する。ボータは同期はずれイベントを検出し、続行する処理セット１０を任意に選別し、処理セット１１および１２をアイドル状態にする。処理セット１０、１１、および１２のそれぞれは、それ自体の主ダーティＲＡＭおよび第２ダーティ・ページ記録を有し、その全てが、同期はずれイベント以降に特異にダーティにされたデータを捕捉する。

【００６３】処理セット１１を再統合するために、処理セット１０のダーティＲＡＭ、処理セット１０の第２ダーティ・ページ記録、処理セット１１のダーティＲＡＭ、および処理セット１１の第２ダーティ・ページ記録においてダーティであると挙げられた全てのページを、処理セット１０から処理セット１１までコピーする。

【００６４】その後、処理セット１２を再統合するために、処理セット１０のダーティＲＡＭ、処理セット１０の第２ダーティ・ページ記録、処理セット１２のダーティＲＡＭ、および処理セット１２の第２ダーティ・ページ記録においてダーティであると挙げられた全てのページを、処理セット１０から処理セット１２までコピーする。

【００６５】処理セット１１の再統合中、処理セット１０はページのダーティ化を記録し続けなければならない。処理セット１０のダーティ・ページＲＡＭがこの過程中のいずれかの部分で動作不可能になった場合、ソフトウェアは、ダーティになり、処理セット１２にコピーされるページのリストに追加されるページの、独立したリストを維持しなければならない。

【００６６】したがって、主ダーティＲＡＭ、例えば制御機構を備えた、専用メモリ管理ユニットまたは従来のメモリ管理ユニットと、主ダーティＲＡＭがイネーブルになるまで全ての不一致書込みを捕捉するのに十分な、同期はずれの前後のメモリへの限られた数の書込みイベントを記録する第２ダーティ・ページ記録と、同期はずれイベントのすぐ後で、主ダーティＲＡＭにページのダーティ化の記録を開始させ、該当する場合は第２ダーテ

ィ・ページ記録を停止させる、ソフトウェアまたはハードウェアどちらかの機構とを備えた、本発明の実施形態について記述した。

【0067】本明細書では、本発明の特定の実施形態について記述したが、本発明の精神および範囲内で、多くの修正および／または追加を加えることができることは理解されるであろう。

【0068】例えば、前述した第1および第2記録機構の様々な組合せを実現することができる。前述した様々なエレメントおよび技術もまた、適当な任意のハードウェアまたはソフトウェア技術を使用して、実施することができる。

【0069】TMRシステムの特定の例について記述したが、本発明はこれに限定されるものではない。さらに、同期はずれ処理セットを識別するために、多数決以外の方法を利用することができる。

【0070】本発明の特定の実施形態について説明したが、本発明はこれらに限定されるものではなく、添付の特許請求の範囲で定義する本発明の精神および範囲内で、多くの修正および／または追加を行うことができることを理解されたい。例えば、従属の特許請求の範囲の特徴の様々な組合せを、独立の特許請求の範囲の特徴と組み合わせることができる。

【図面の簡単な説明】

【図1】モジュラ三重化フォールト・トレラント・コンピュータ・システムを示す概観図である。

【図2】図1のシステムの処理セットのエレメントを示す概略図である。

【図3】本発明の一実施形態の処理セットを示す概略図である。

【図4】メモリ管理ユニットを示す概略図である。

【図5】第1記録機構の一例を示す概略図である。

【図6】第2記録機構の一例を示す概略図である。

【図7】第2記録機構の別の例を示す概略図である。

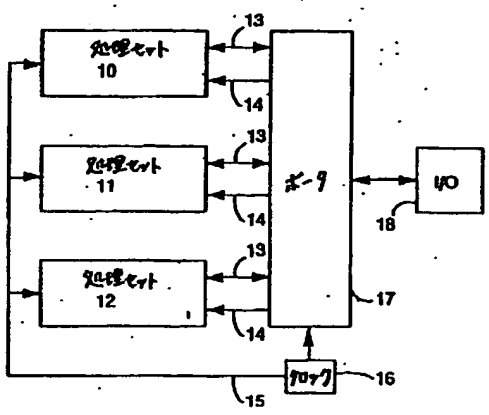
【図8】合成された第1および第2記録機構の一例を示す概略図である。

【図9】図8の例の代替構成を示す概略図である。

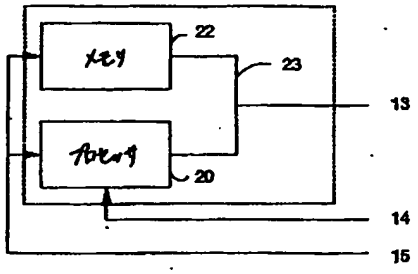
【符号の説明】

10／11／12　処理セット
13　内部バスからの接続
14　ハードウェア割込み入力
15　外部クロックからの入力
20　プロセッサ
22　メモリ
23　内部バス
25　第1記録機構
26　第2記録機構
27　再統合機構

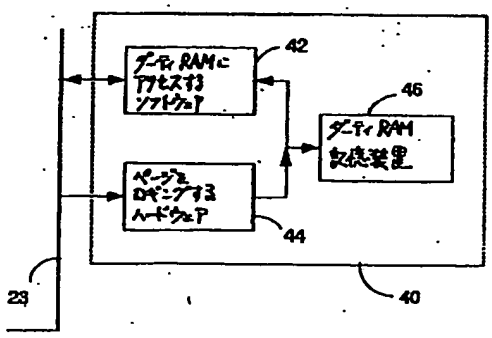
【図1】



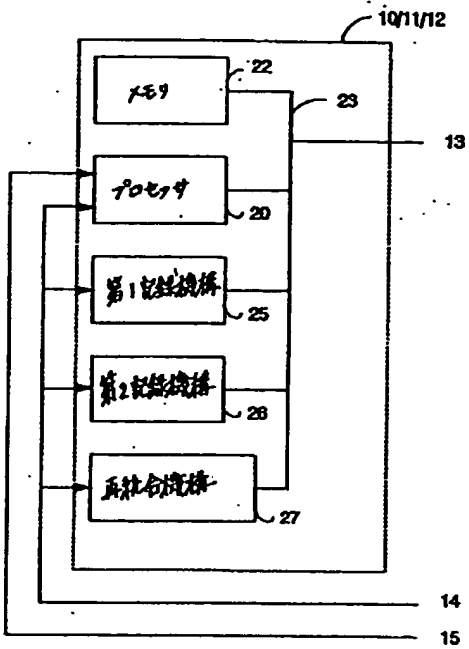
【図2】



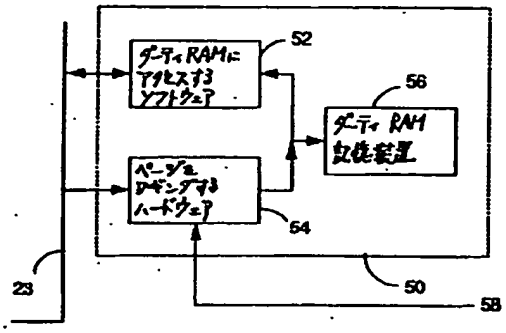
【図4】



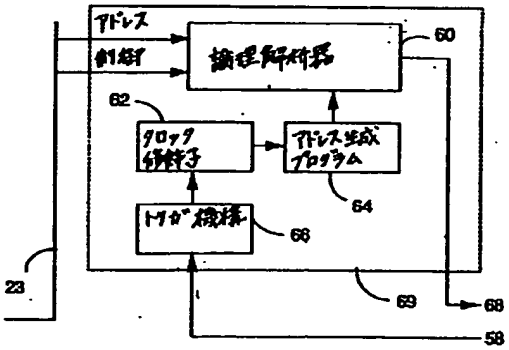
【図3】



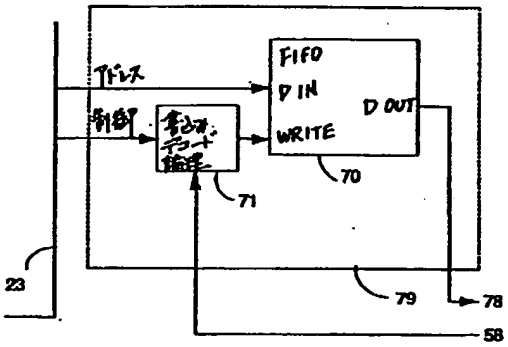
【図5】



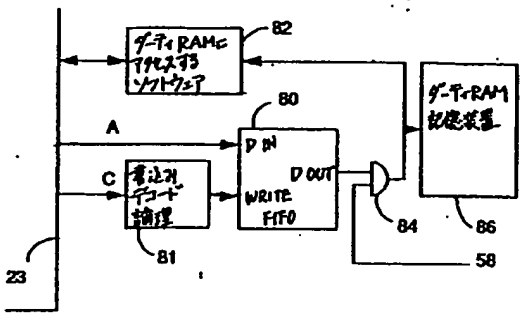
【図6】



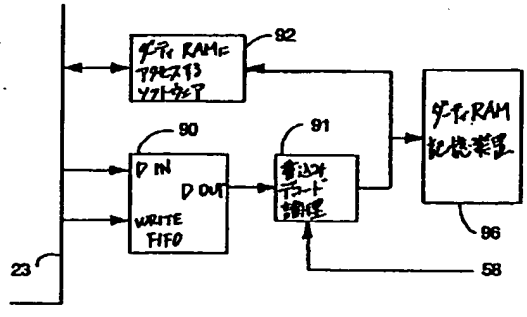
【図7】



【図8】



【図9】



フロントページの続き

(71)出願人 591064003
901 SAN ANTONIO ROAD
PALO ALTO, CA 94303, U.
S. A.

(72)発明者 エムリス・ジョン・ウイリアムズ
イギリス国・エムケイ17 9ディエス・ミルトン キーンズ・エヴァーショルト・ティレルズ エンド・ヘルフォード ハウス（番地なし）

【外国語明細書】

MEMORY MANAGEMENT IN FAULT TOLERANT COMPUTER SYSTEMS

BACKGROUND AND INTRODUCTION

The invention relates generally to fault tolerant computer systems such as lockstep fault tolerant computers which use multiple subsystems that run identically.

In such lockstep fault tolerant computer systems, the outputs of the subsystems are compared within the computer and, if the outputs differ, some exceptional repair action is taken.

Figure 1 of the accompanying drawings is a schematic overview of an example of a typical system, in which three identical processing (CPU) sets 10, 11, 12 operate in synchronism (sync) under a common clock 16. By a processing set is meant a subsystem including a processing engine, for example a central processing unit (CPU), and internal state storage. Figure 2 of the accompanying drawings is a schematic representation of such a processing set. This shows a processing engine 20, internal state storage (memory) 22 and an internal bus 23. The processing set may include other elements of a computer system, but will not normally include input/output interfaces. External connections are also provided, for example a connection 13 from the internal bus 13, an input 15 for the external clock 16 and hardware interrupt inputs 14.

As shown in Figure 1, the outputs of the three processing sets 10, 11, 12 are supplied to a fault detector unit (voter) 17 to monitor the operation of the processing sets 10, 11, 12. If the processors sets 10, 11, 12 are operating correctly, they produce identical outputs to the voter 17. Accordingly, if the outputs match, the voter 17 passes commands from the processing sets 10, 11, 12 to an input/output (I/O) subsystem 18 for action. If, however, the outputs from the processing sets differ, this indicates that something is amiss, and the voter causes some corrective action to occur before acting upon an I/O operation.

Typically, a corrective action includes the voter supplying a signal via the appropriate line 14 to a processing set showing a fault to cause a "change me" light (not shown) to be illuminated on the faulty processing set. The defective processing set is switched off and an operator then has to replace it with a correctly functioning unit. In the example shown, a defective processing set can normally be easily identified by majority voting because of the two-to-one vote that will occur if one

processing set fails or develops a temporary or permanent fault.

However, the invention is not limited to such systems, but is also applicable to systems where extensive diagnostic operations are needed to identify the faulty processing set. The system need not have a single voter, and need not vote merely I/O commands. The invention is generally applicable to synchronous systems with redundant components which run in lockstep.

Lockstep systems depend on total synchronisation of the processing sets that make up the fault tolerant processing core. Accordingly, the processing sets need hardware which operates identically, and, in addition, the internal stored state of the data in the processing sets also needs to be identical. Part of the process of integrating a new processing set into a running system involves copying the contents of the main memory of a running system to the new processing set. Because main memory can be very large, for example of the order of gigabytes, this process can take rather a long time in computing terms.

Lockstep computer systems can go out of sync for various reasons. The prime reason is a failure of a single processing set in a permanent way. Recovery from such a failure normally involves removal of the failed unit, replacement with a functioning unit and reinstatement of the functioning unit. Clearly, the new processing set will have no notion of the contents of memory of a running processing set, and all of the main memory from the running system will have to be copied to the new processing set.

Other, less traumatic out-of-sync events can often be diagnosed automatically by the running computer system and can lead to the automatic reintegration of the out-of-sync processing set without its replacement. For example, a soft data error in a dynamic memory, perhaps caused by a cosmic ray event, could cause a minor upset in operation that could be fixed automatically. However, this has still required the reintegration of the memory state of the out-of-sync processing set, that is the copying of the contents of the main memory from a running system to the out-of-sync processing set. Accordingly, because of the main memory can be very large, this can still take a long time in computing terms.

The invention seeks to provide an automatic and rapid way of recovering from minor out-of-sync events which avoids the problems of the prior art.

SUMMARY OF THE INVENTION

In accordance with one aspect of the invention, there is provided a memory management system for a fault tolerant computer system, the memory management system comprising: a first recording mechanism which can be activated to record memory update events; a second recording mechanism having a capacity to record at least a limited number of memory update events; a fault input for a fault signal to activate the first recording mechanism in the event of a fault event; and a memory reintegration mechanism to reintegrate at least parts of memory identified in the first and second recording mechanisms.

Embodiments of the invention take advantage of the fact that, after a minor out-of-sync event between processing sets in a lockstep system, most of the memory contents of the out-of-sync processing set is initially identical to that in a running system. Only a relatively small number of locations within the memory system of either the out-of-sync processing set or the running system will have been modified. However, the divergence will increase with time as the running system continues to operate and execute its normal processing load. Embodiments of the invention allow for the divergence to be tracked and accounted for and, moreover, for any memory update events around the out-of-sync event and before the first recording mechanism has been activated to be caught.

Preferably, the recording of memory updates (writes) is not based on recording each address accessed, but rather on memory segments (pages) updated (written to). In other words, the first and/or second recording mechanisms preferably record the segments (or pages) updated (written to). This can be done effectively using a segment (or page) memory with a bit per segment (page) for identifying the segments (pages) written to.

In accordance with another aspect of the invention, there is provided a fault tolerant computer system comprising a plurality of synchronous processing sets, each comprising a processor with internal memory and operating in lockstep, and an out of sync detector for detecting an out-of sync-event and for generating an out-of-sync signal, wherein each processing set also comprises: a first recording mechanism which can be activated to record memory write events; a second recording mechanism

having a capacity to record at least a limited number of memory write events; a fault input for receiving the out-of-sync signal to activate the first recording mechanism in the event of an out-of-sync event; and a memory reintegration mechanism to reintegrate in an out-of sync processing set at least parts of memory identified in the first and second recording mechanisms.

In accordance with a further aspect of the invention, there is provided a method for reintegration of a processing set of a fault tolerant computer system following a fault, wherein the fault tolerant computer system comprises a plurality of synchronous processing sets, each comprising a processor and internal memory and operating in lockstep, and a fault detector for detecting a fault event and for generating a fault signal, the method comprising:

maintaining a temporary record of memory update events over a limited period;

responding to a fault to activate a further record of memory update events following the fault state; and

performing memory reintegration in a processing set in which a fault has occurred for at least those parts of memory identified in the temporary and further memory records.

In an embodiment of the invention a record is kept of at least selected memory access events (memory write events) to main memory after the out-of-sync event, so that only the modified memory locations need to be copied to reintegrate the out-of-sync processing set.

In one embodiment of the invention, the first recording mechanism is a memory management unit comprising a RAM with an entry for each of a plurality of memory pages, a code being written to a page entry each time that page is written to when the first recording mechanism has been activated.

Preferably, the first recording mechanism has an enable input connected to received the fault (out-of-sync) signal.

The first recording mechanism can record an arbitrarily large number of written pages, up to the total number of pages in the processing unit.

The second recording mechanism preferably maintains a rolling record of recent memory update events up to a number sufficient to cover the time to activate the first recording mechanism following a fault event.

The second recording mechanism can comprise a first-in-first out buffer, the first recording mechanism in one embodiment being connected to an output of the first-in-first-out buffer. In this configuration, the first-in-first-out buffer stores up to a predetermined number of update addresses, an address decoder can be connected to the output of the first-in-first-out buffer to generate a page signal representative of a memory update address output from the first-in-first-out buffer and the address decoder is responsive to the out-of sync signal to pass the page signal to the first recording mechanism.

Alternatively, the second recording mechanism can comprise a logic analyzer. This can reduce implementation costs as fault tolerant computers typically include a logic analyzer for fault analysis.

Where the output of the second recording mechanism does not form the input to the first recording mechanism, the operation of the second recording means is preferably inhibited in response to the out-of-sync signal.

The first recording mechanism can comprise a software generated table in which a record corresponding to a page of memory is marked with a code whenever that page has been written. This record can be maintained by software which updates entries in the translation look-aside buffer of the processor. The second recording mechanism can be the contents of the TLB together with a list of pages recently flushed from the TLB.

In response to an out-of-sync input, software can search the TLB and the list for pages which may recently have been written and may mark these as written in the first recording mechanism, then continue to maintain the first recording mechanism until the processing units are reintegrated.

Preferably, the memory reintegration mechanism is operative to reintegrate memory pages identified in the first and second recording mechanisms.

The invention is applicable to a computer system comprising three synchronous processing sets operating in lockstep, wherein an out-of-sync detector determines an out-of-sync processing set by majority voting.

In this case, reintegration of an out-of-step processing set can be achieved by, in response to the identification of an out-of-sync processing set, selecting one of the remaining two processing sets, supplying an interrupt to the out-of-sync processing

set and the remaining processing set to cause the out-of-sync and remaining processing sets to idle, reintegrating one of the out-of-sync and remaining processing sets while maintaining a software log of memory write events, and then reintegrating the other of the out-of-sync and remaining processing sets using the software log.

The invention described can reduce the reintegration time of a processing set in a lockstep fault tolerant computer from many minutes to just fractions of a second. During the reintegration period, the computer is vulnerable to a further failure in the running processing set. Thus the reduction in reintegration time has a significant benefit on the overall availability of the computer.

DESCRIPTION OF THE DRAWINGS

An embodiment of the invention will be described hereinafter with reference to the accompanying drawings in which like reference signs relate to like features and in which:

Figure 1 is a schematic overview of a triple-modular-redundant fault tolerant computer system;

Figure 2 is a schematic representation of elements of a processor set of the system of Figure 1;

Figure 3 is a schematic representation of a processor set of an embodiment of the invention;

Figure 4 is a schematic representation of a memory management unit;

Figure 5 is a schematic representation of an example of a first recording mechanism;

Figure 6 is a schematic representation of an example of a secondary recording mechanism;

Figure 7 is a schematic representation of another example of a secondary recording mechanism;

Figure 8 is a schematic representation of an example of a combined first and secondary recording mechanism; and

Figure 9 is a schematic representation of an alternative configuration of the example of Figure 8.

DESCRIPTION OF THE PREFERRED EMBODIMENT

Figure 3 is a schematic block diagram to represent elements of an example of the invention in functional terms. Figure 3 generally represents one of the processing sets 10/11/12 for a fault tolerant computer system such as, for example, the system shown in Figure 1.

In Figure 3, a processing engine (e.g. a central processing unit (CPU)) 20 and internal state storage (memory) 22 are connected by an internal bus 23. External connections are also provided, for example a connection 13 from the internal bus 23, an input 15 for an external clock and hardware interrupt input 14.

Also shown schematically in Figure 3 are a first recording mechanism 25 which can be activated to record memory update events, a second recording mechanism 26 having a capacity to record at least a limited number of memory update events and a memory reintegration mechanism to reintegrate at least parts of memory identified in the first and second recording mechanisms. As shown in Figure 3, each of the mechanisms 25, 26 and 27 is shown connected to the internal bus 23. This is because the first and second recording mechanisms 25 and 26 need to monitor memory access events to identify when and where memory updating occurs (memory writes). Also the reintegration mechanism 27 needs to access the first and second recording mechanisms to determine where memory writes have occurred in the memories of out-of-sync and running processing sets and then to copy corresponding memory portions from the running to out-of-sync processor set memories. However, the mechanisms 25, 26 and 27 can be implemented in various ways as will be explained in the following description. Various implementations involve different combinations of hardware and software and interconnection of the various elements will typically differ from that illustrated in Figure 3. For example, the reintegration mechanism will typically be implemented in software, and may be implemented in a control computer associated with and/or forming part of the voter 17. Also, the first recording mechanism 25 may not be connected directly to the bus 23, but may be connected via the second recording mechanism 26. Also the first and second recording mechanisms may be implemented to a greater or lesser extent in software, as will be described hereinafter.

Computer systems typically include memory management hardware to keep

track of and control the use of main memory. It is also usual to divide memory into pages of specified size and to keep a small record of access controls to each page. Hardware mechanisms also exist for updating a record for a page with a bit that indicates that the page has been modified. This bit is the so-called 'dirty' bit for a page. A page of memory is called 'clean' when no writes have been made to it to change it from its initial state, and 'dirty' after such a write has been made. Software can cause a page to be marked 'clean' by clearing the dirty bit for that page in the memory management unit record. Hardware will later set the bit to 1 to indicate that the page has been written to. In normal operation, many pages of computer memory will be considered by the memory management unit to be dirty most of the time. Accordingly, if a conventional memory management unit operating in a conventional manner is provided in each of the processing sets of a lockstep fault tolerant computer system, it is thus likely that many pages will be marked dirty when an out-of-sync event occurs.

Because the memory management unit of a conventionally configured computer processing set is usually under the control of the running operating system, in a first embodiment of the invention an additional memory management unit is provided purely for the use of the software which reintegrates processing sets after an out-of-sync event.

Figure 4 shows a conventional memory management unit 40 which has been customised to include only information on which pages of memory are dirty and which are clean. In the following description this type of memory management unit is termed a 'dirty ram'. Software 42 may access the dirty ram storage 46 to check which pages are dirty, and can write it directly to change the status of a page to dirty or clean. In addition, hardware 44 automatically changes to 'dirty' the state of the record for any page of main memory which is written to via the bus 23. In this embodiment only one bit of dirty ram storage 46 is used for each entire page of main memory. It is not necessary that the size of the 'pages' monitored by the dirty ram is the same as the size used by other memory management units in the system, but it is often both convenient and efficient that the pages all have the same size. Computers tend to work in pages and a write access to one part of a page often implies that other locations within the same page will also be written. However, a

conventional memory management unit as shown in Figure 3 will not in itself be sufficient to implement the task in hand because most of the pages are usually dirty as described above.

Figure 5 is a schematic block diagram of a dirty ram 50 for a first embodiment of the invention. In Figure 5, the dirty ram 50 is provided with a separate enable input 58 whereby the hardware 54 only begins to log dirty pages in the dirty ram storage 56 after the processing sets have gone out of sync. The signal on the enable input is asserted in response to the detection by the voter 17 of an out-of-sync event.

The dirty ram enable input 58 allows the operation of the dirty ram system. In normal operation, with processing sets running in sync, the dirty ram enable input is not asserted and the dirty ram 50 is set by the software 52 such that all pages are given 'clean' status.

When an out-of-sync event occurs, the enable input 58 becomes asserted. While the enable input is asserted (i.e., while the processing sets are out of sync), the dirty ram logs the pages of main memory written to. The pages which are written to will be those which potentially differ on the running and the out-of-sync processing sets. A dirty ram with an enable input as in Figure 5 is provided in each processing set and is connected there to the respective system bus 23. While the processing sets are running in sync, each dirty ram is held in the clean state by the software 52. When it is detected that the processing sets are running out of sync the dirty ram logging is enabled. In this embodiment, a hardware enable signal 58 is generated in the out-of-sync detection hardware (i.e., the voter 17) on detecting that the processing sets 10, 11, 12 are out-of-sync. In other words, whenever the voter detects a difference in the output from the processing sets, it generates a signal which is supplied to each processing set to form the asserted out-of-sync signal. Once asserted, the out-of-sync signal is not negated until the processing sets have been reinstated. In other embodiments, the enable signal could be generated by software.

After an out-of-sync event, software and/or hardware mechanisms act to re-configure the fault tolerant computer system. The system carries on running normal operations with at least one processing set. At least one processing set, including the out-of-sync processing set is taken out of operation. This out-of-sync processing set

thereby stops running normal perations and waits to be reintegrated into the running system. Memory writes done on the running and the out-of-sync processing set produce divergence in the main memory contents in the running and out-of-sync processing sets.

When software on the running system comes to reintegrate the out-of-sync processing set, it accesses the dirty ram on the running system to find pages of memory that have been dirtied since the out-of-sync event. It also accesses the dirty ram in the out-of-sync processing set. This dirty ram tells which pages have been modified by the out-of-sync processor(s) since the divergence began. If a page of memory is mentioned as dirty in any of the dirty rams, on the running and out-of-sync processing sets, it has to be copied by the reintegration software to bring the out-of-sync processing sets back into sync. If a page of memory is not marked as dirty in any dirty ram, it can be ignored, as it will still be correct on the out-of-sync processing set.

In an alternative embodiment of the invention, if the processing sets have a dirty ram store with no enable pin, operating all the time to log dirty pages, software could be activated by a hardware signal on the out-of-sync event to clean out the dirty ram. This software must carefully note any pages which it itself dirties during the cleaning process.

In yet another alternative embodiment, an ordinary memory management unit can also be used to collect the dirty page information. In this alternative embodiment, software is arranged to modify the page tables at the out-of-sync event so that all pages of main memory are write protected. This means that write cycles to memory will result in a bus error exception to the processor. The processor can then act on each bus error first to add the written page to a software-maintained list of dirty pages, then to remove the write protection for that page so that future writes there will complete normally. This has the advantage that only a single list of dirty pages need be examined by the reintegration software, with no searching through clean pages to look for occasional dirty ones.

It should be noted, however, that it is desirable to provide a separate 'dirty memory' rather than to use using conventional memory management units with additional software to collect dirty page information. This is because the use of

conventional memory management units suffers from twin disadvantages. Firstly, the conventional computer operating system software may be using the memory management unit for its own purposes. Secondly, conventional memory management units work for only a single processor and cannot cover multi-processor operation or direct memory access by I/O devices.

Whatever method is used to collect the data on dirty memory pages, there are likely to be problems near the out-of-sync event. Some time has to elapse between the detection of the out-of-sync event and the enabling of the dirty ram data collection, and a few dirty pages may go unrecorded in this period. Exactly how many pages are missed depends on the implementation of the dirty ram, but even a single missed page is enough to make useless the scheme of copying only some, not all, of main memory after an out-of-sync event.

Accordingly, in embodiments of the invention, a mechanism is also provided for recording memory write events around the out-of-sync event.

The mechanisms described above for providing a record of dirty pages are put into operation starting at the out-of-sync event, and can record all the pages required following that event. However, to complement this a separate, temporary, record is required for pages dirtied close to the out-of-sync event. This separate record has to take account of write events over a limited time, preferably on a rolling basis. This separate record needs to have a capacity sufficient to accommodate write events which might occur between the out-of sync event itself and the time the mechanisms described above can start recording. This separate record is called the secondary dirty page record in the following description, to distinguish it from the 'dirty ram' already discussed above.

The secondary dirty page record has to be operating continually (at least until an out-of-sync event occurs), because it cannot be predicted when a fault will send it out-of-sync. It is the job of the secondary dirty page record to remember those pages, dirtied just before or after an out-of-sync event, that may not be properly collected by the dirty ram. The secondary dirty page record also has to have limited time memory. If it remembered pages dirtied indefinitely far in the past, it would eventually list all memory pages as being dirty. It should only remember far enough back past the out-of-sync event to ensure that divergently dirtied pages which the

primary dirty page store cannot catch are included.

In some embodiments described below, where the secondary dirty pag record operates in parallel with the dirty ram, the secondary page record is frozen at or soon after the out-of-sync event. In these embodiments, if it were left running, the limited-time nature of the record could eventually allow the important information about the out-of sync event to be overwritten or lost. This can conveniently be done by responding to the asserting of the dirty ram enable signal to inhibit the operation of the secondary page record.

Once the secondary dirty page record has been frozen, either software or hardware can examine it and cause dirty pages listed there to be added to the primary dirty ram, or to a separate list for copying by the reintegration software. Both out-of-sync and running processing sets have secondary dirty page records. Software can examine and compare the records and deduce which pages were actually dirtied in sync by the processing sets, if needed. This will decrease the number of pages to be copied.

In one embodiment a logic analyzer is used to collect information for the secondary dirty page record. Figure 6 shows a logic analyzer 60 observing the bus 23 of a processing set. A logic analyzer 60 with a trigger mechanism 66, clock qualifier 62 and address generator 64, is provided for each processing set. The logic analyzer is usually running. The analyzer 60 is triggered causing data collection to stop, by the assertion of the processing set out-of-sync signal, which same signal starts the primary dirty ram collecting data. The logic analyzer eventually stops operating and keeps a record of computer bus operation both before and after the out-of-sync event. By analysing the logic analyzer traces from the out-of-sync and running processing sets after an out-of-sync event, it is possible to deduce which of the stored transactions is a divergent write cycle. The relevant pages can then be added to a set of pages written to in a software-maintained secondary dirty page record. The logic analyzer needs to store at least the address and control information for each bus cycle so that pages written to can be determined. The analysis of the logic analyzer outputs can readily be effected by software routines.

The advantage of using a logic analyzer for the secondary dirty page record is that lockstep fault tolerant computers will typically have logic analyzers built in,

and triggered on the out-of-sync event, for fault diagnosis. It is then merely necessary to provide the software to control and analyze the output of the logic analyzers as described above.

As an alternative, a write buffer can provide storage for the secondary dirty page record. Figure 7 shows a first-in-first-out memory 70 used as a short-term buffer for writes to main memory over the internal bus 23. In normal in-sync operation, writes to main memory are decoded by write decode logic 71 and the page number of each write is written into the FIFO 70. When the out-of-sync occurs, the hardware out-of-sync detection signal 58 inhibits further writes into the FIFO 70. Later, software can examine the FIFO 70 contents to add pages to the dirty page list. The advantage of the write buffer for a secondary dirty page record is that it is simpler in both software and hardware than a logic analyzer.

In further alternatives, the write buffer can be arranged in series with the dirty ram.

Figure 8 illustrates a first example of a combination of a first and secondary recording mechanism, where a write buffer is arranged in series with a dirty ram. The arrangement of Figure 8 is based on a combination of the arrangements of Figures 4 and 7. In this case, a FIFO buffer 80 is operating continually to record write events to memory for a processing set with the write events being decoded continually by the write decode logic 81 to supply page addresses for storage in the FIFO 80. In the present case it is not necessary for the write decode logic to receive an inhibit input as will be described later. Page addresses supplied to the FIFO 80 appear, after a delay, at the output of the FIFO 80. However, they are prevented from being passed to the dirty ram storage 86 by means of the gate 84 until the gate is enabled by an out-of-sync signal on the line 58. This out-of sync signal effectively provides an enable signal for the dirty ram as it then enables the page addresses from the FIFO 80 to be supplied to the dirty ram storage 86, whereby appropriate page bits can be set. Software 82 can be used to clear the dirty ram storage 86 at any time prior to an out-of-sync event so that it is 'clean' when the out-of-sync signal is supplied.

In this embodiment, it is not necessary to disable the FIFO buffer 80, as the contents of the FIFO buffer 80 are automatically stored in the dirty ram after a time

dependent on the size f the FIFO 80 and the frequency of the write events. In this embodiment the reintegration mechanism preferably takes account of the dirty ram 86 and the FIFO 80.

Figure 9 illustrates a second example of a combination of a first and secondary recording mechanism, where a write buffer is arranged in series with a dirty ram. In this example, a FIFO buffer 90 is operating continually to record write events to memory for a processing set. The output from the FIFO buffer 90 is supplied to an address decoder 91 which is only enabled in response to the out-of-sync enable signal 58. When the address decoder 91 is not enabled, the output of the FIFO buffer is effectively discarded. Only when the address decoder is enabled are dirty page bits output from the address decoder 91 for storage in the appropriate page location in the dirty ram storage 96.

Optionally, the out-of-sync enable signal is also supplied to the dirty ram itself, although it will be appreciated that supplying the out-of-sync signal to the address decoder effectively provides an enable signal for the dirty ram. As for the Figure 8 example, in this embodiment it is not necessary to disable the FIFO buffer 90, as the contents of the FIFO buffer 90 are automatically stored in the dirty ram after a time dependent on the size of the FIFO 90 and the frequency of the write events. In this embodiment the reintegration mechanism preferably takes account of the dirty ram 96 and the FIFO 90.

A further, software-implemented embodiment makes use of a table look-aside buffer (TLB) and an associated TLB miss routine, plus a dirty page store which is created in main memory. A TLB forms a standard part of most computer addressing schemes using paged memory. Some computers maintain TLB entries with a software TLB miss routine, instead of fixed hardware. In this embodiment, and following a fault event, software can note the TLB entries currently specifying writable pages, and add those to the soft dirty page store. Software can also transfer to this a list of writable pages recently flushed from the TLB. This list is maintained

by the miss routine in normal circumstances, before the fault event, and indicates pages which might have been written close to the fault event and immediately flushed from the TLB. Following this, while reintegration is in progress, software in the TLB miss routine adds each page written to the soft dirty page store. In this way, a record of dirty pages can be created.

The approaches described above can be applied in a triple-modular-redundant (TMR) system. Some TMR lockstep systems switch to running with a single processing set when an out-of-sync event occurs. This requires two separate reintegration phases to recover.

An example will be described where a TMR system is running with processing sets 10, 11 and 12 in sync. In this example, reintegration is performed by software under the control of a control computer forming part of the voter 17.

For this example, it is assumed here that processing set 12 suffers a dram soft error which takes the system out of sync. The voter detects the out-of-sync event and arbitrarily chooses processing set 10 to carry on and idles processing sets 11 and 12. Each of processing sets 10, 11 and 12 has its own primary dirty ram and secondary dirty page record, all capturing the differentially dirtied data since the out-of-sync event.

To reintegrate processing set 11, all of the pages mentioned as dirty in processing set 10 dirty ram, processing set 10 secondary dirty page record, processing set 11 dirty ram and processing set 11 secondary dirty page record are copied from processing set 10 to processing set 11.

Then, to reintegrate processing set 12, all the pages mentioned as dirty in processing set 10 dirty ram, processing set 10 secondary dirty page record, processing set 12 dirty ram and processing set 12 secondary dirty page record are copied from processing set 10 to processing set 12.

During the reintegration of processing set 11, the processing set 10 has to continue to record the dirtying of pages. If the processing set 10 dirty page ram is inoperative during some part of this process, software must maintain a separate list of pages being dirtied to add to the list of pages copied to processing set 12.

There has been described, therefore, embodiments of the invention in which there are provided: a primary dirty ram, for example a dedicated memory

management unit or a conventi nal memory management unit with control mechanisms; a secondary dirty page record which records a limited number of write events to memory around the out-of-sync event, sufficient to capture all divergent writes until the primary dirty ram is enabled; and a mechanism, either hardware or software, to start the primary dirty ram recording the dirtying of pages shortly after an out-of-sync event, and where appropriate to stop the secondary dirty page record. By examining the primary dirty ram and secondary dirty page record of two processing sets and copying the pages of memory mentioned as dirty in any dirty page record to the out-of-sync processing set the system can be reinstated in an efficient manner.

Although particular embodiments of the invention have been described herein, it will be appreciated that many modifications and/or additions may be made within the spirit and scope of the present invention.

For example, various combinations of the first and second recording mechanisms described above may be provided. Also the various elements and techniques described above may be implemented using any appropriate hardware or software technology.

Although a specific example of a TMR system has been described, the invention is not limited thereto. Moreover, other methods than majority voting can be used to identify an out-of-sync processing set.

Although particular embodiments of the inventions have been described, it will be appreciated that the invention is not limited thereto, and many modifications and/or additions may be made within the spirit and scope of the invention as defined in the appended Claims. For example, different combinations of the features of the dependent Claims may be combined with the features of the independent Claims.

WHAT I CLAIM IS:

1. A memory management system for a fault tolerant computer system, said memory management system comprising:
a first recording mechanism which can be activated to record memory update events;
a second recording mechanism having a capacity to record at least a limited number of memory update events;
a fault input for a fault signal to activate said first recording mechanism in the event of a fault event; and
a memory reintegration mechanism to reintegrate at least parts of memory identified in said first and second recording mechanisms.

2. A memory management system according to Claim 1, wherein said first recording mechanism is a memory management unit comprising storage having an entry for each of a plurality of memory pages, a code being written to a page entry each time that page is written to when said first recording mechanism has been activated.

3. A memory management system according to Claim 2, wherein said first recording mechanism has an enable input connected to receive the fault signal for activating said first recording mechanism.

4. A memory management system according to Claim 1, wherein said second recording mechanism maintains a record of recent memory update events up to a number sufficient to cover the time to activate said first recording mechanism following a fault event.

5. A memory management system according to Claim 4, wherein said second recording mechanism comprises a first-in-first-out buffer.

6. A memory management system according to Claim 5, wherein said first recording mechanism is connected to an output of said first-in-first-out buffer.

7. A memory management system according to Claim 6, wherein said first-in-first-out buffer stores up to a predetermined number of update addresses, an address decoder is connected to said output of said first-in-first-out buffer to generate a page signal representative of a memory update address output from said first-in-first-out buffer and said address decoder is responsive to said fault signal to pass said page signal to said first recording mechanism.

8. A memory management system according to Claim 1, wherein said second recording mechanism comprises a logic analyzer.

9. A memory management system according to Claim 1, wherein said second recording mechanism maintains a record of recent memory update events up to a number sufficient to cover the time to activate said first recording mechanism following a fault event, the operation of said second recording means being inhibited in response to said fault signal.

10. A memory management system according to Claim 9, wherein said first recording mechanism comprises a software generated list of update events.

11. A memory management system according to Claim 1, wherein said second recording mechanism comprises a table look-aside buffer and a memory access table maintained in main memory.

12. A memory management system according to Claim 1, wherein said memory reintegration mechanism is operative to reintegrate memory pages identified in said first and second recording mechanisms.

13. A fault tolerant computer system comprising a plurality of synchronous processing sets, each comprising a processor and internal memory and operating in lockstep, and an out-of-sync detector for detecting an out-of-sync event and for generating an out-of-sync signal, wherein each processing set includes a memory management system according to Claim 1.

14. A fault tolerant computer system comprising a plurality of synchronous processing sets, each comprising a processor and internal memory and operating in lockstep, and an out-of-sync detector for detecting an out-of-sync event and for generating an out-of-sync signal, wherein each processing set also comprises:
a first recording mechanism which can be activated to record memory write events;
a second recording mechanism having a capacity to record at least a limited number of memory write events;
a fault input for receiving said out-of-sync signal to activate said first recording mechanism in the event of an out-of-sync event; and
a memory reintegration mechanism to reintegrate in an out-of-sync processing set at least parts of memory identified in said first and second recording mechanisms.

15. A fault tolerant computer system according to Claim 14, wherein said first recording mechanism is a memory management unit comprising a RAM with an entry for each of a plurality of memory pages, a code being written to a page entry each time that page is written to when said first recording mechanism has been activated.

16. A fault tolerant computer system according to Claim 15, wherein said first recording mechanism has an enable input connected to receive the out-of-sync signal for activating said first recording mechanism.

17. A fault tolerant computer system according to Claim 14, wherein said second recording mechanism maintains a record of recent memory update events up to a number sufficient to cover the time to activate said first recording mechanism following an out-of-sync event.

18. A fault tolerant computer system according to Claim 17, wherein said second recording mechanism comprises a first-in-first-out buffer.

19. A fault tolerant computer system according to Claim 18, wherein said first recording mechanism is connected to an output of said first-in-first-out buffer.

20. A fault tolerant computer system according to Claim 19, wherein said first-in-first-out buffer stores up to a predetermined number of update addresses, an address decoder is connected to said output of said first-in-first-out buffer to generate a page signal representative of a memory update address output from said first-in-first-out buffer and said address decoder is responsive to said out-of-sync signal to pass said page signal to said first recording mechanism.

21. A fault tolerant computer system according to Claim 14, wherein said second recording mechanism comprises a logic analyzer.

22. A fault tolerant computer system according to Claim 14, wherein said second recording mechanism maintains a record of recent memory update events up to a number sufficient to cover the time to activate said first recording mechanism following an out-of-sync event, the operation of said second recording means being inhibited in response to said out-of-sync signal.

23. A fault tolerant computer system according to Claim 22, wherein said first recording mechanism comprises a software generated list of update events.

24. A fault tolerant computer system according to Claim 14, wherein said second recording mechanism comprises a table look-aside buffer and a memory access table maintained in main memory.

25. A fault tolerant computer system according to Claim 14, wherein said memory reintegration mechanism is operative to reintegrate memory pages identified in said first and second recording mechanisms.

26. A fault tolerant computer system according to Claim 14, comprising three synchronous processing sets operating in lockstep, wherein said out-of-sync detector determines an out-of-sync processing set by majority voting.

27. A fault tolerant computer system according to Claim 26, wherein said out-of-

sync detector is arranged to select one of the remaining two processing sets, to supply an input to the out-of-sync processing set and the remaining processing set to cause said out-of-sync and remaining processing sets to idle, to reintegrate one f said out-of-sync and remaining processing sets while maintaining a software log of memory write events, and then to reintegrate said other of said out-of-sync and remaining processing sets using said software log .

28. A method for reintegration of a processing set of a fault tolerant computer system comprising a plurality of synchronous processing sets, each comprising a processor and internal memory and operating in lockstep, and a fault detector for detecting a fault event and for generating a fault signal, said method comprising:
maintaining a temporary record of memory update events over a limited period;
responding to said fault signal to activate a further record of memory update events following said fault state; and
performing memory reintegration in a processor in which a fault has occurred for at least those parts of memory identified in said temporary and further memory records.

29. A method according to Claim 28, wherein said fault event is an out-of-sync event.

30. A method according to Claim 28, wherein the further record is stored in a storage of a memory management unit, a page entry in said storage being provided for each page of memory, a code being written to a page entry each time that page is written to when said first recording mechanism has been activated.

31. A method according to Claim 28, comprising maintaining a record of recent memory update events up to a number sufficient to cover the time to activate said further record.

32. A method according to Claim 31, wherein said temporary record is stored in a first-in-first-out buffer.

33. A method according to Claim 32, comprising connecting a recording mechanism for the further recording to an output of said first-in-first-out buffer.

34. A method according to Claim 33, comprising the steps of storing up to a predetermined number of update addresses in said first-in-first-out buffer, supplying the output of said first-in-first-out buffer to an address decoder to generate a page signal representative of a memory update address output from said first-in-first-out buffer, and recording said page signal as part of said further recording when said fault signal is active.

35. A method according to Claim 31, wherein said temporary record is stored in a logic analyzer.

36. A method according to Claim 31, comprising the steps of maintaining said temporary record of recent memory update events up to a number sufficient to cover the time to activate said first recording mechanism following a fault, said temporary record being inhibited in response to said fault signal.

37. A method according to Claim 36, comprising generating a list of update events by software.

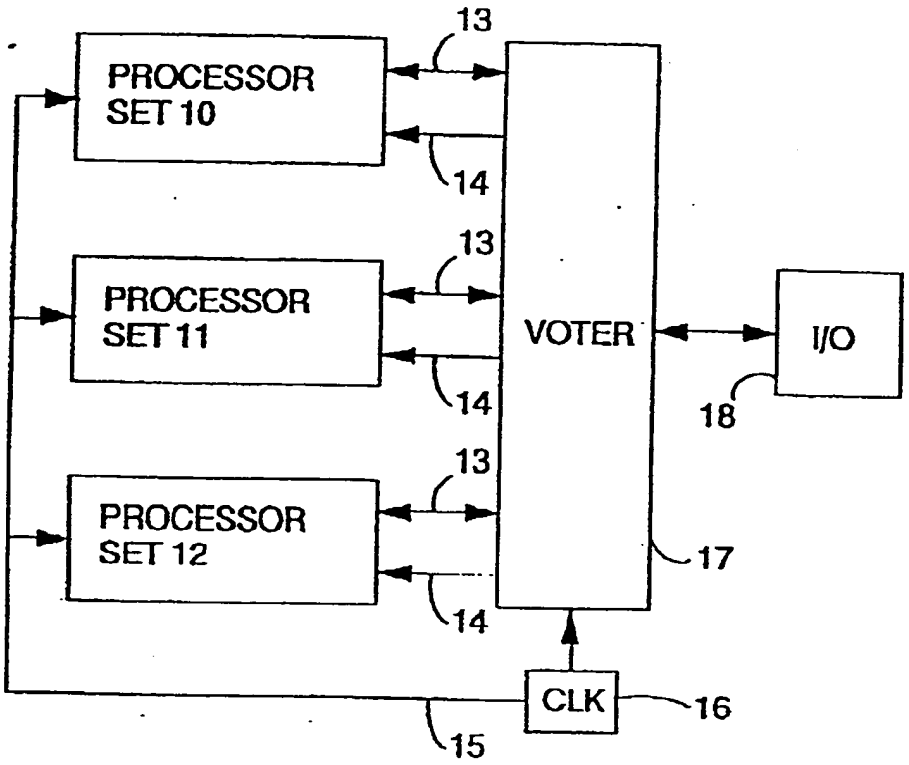
38. A method according to Claim 31, comprising forming said temporary record by means of a table look-aside buffer and a memory access table maintained in main memory.

39. A method according to Claim 31, comprising the step of reintegrating memory pages identified in said temporary and further records.

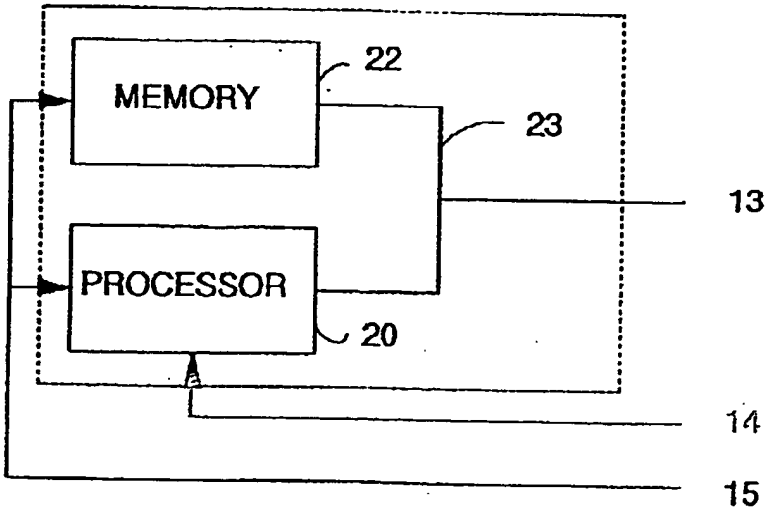
40. A method according to Claim 31, comprising three synchronous processing sets operating in lockstep, wherein an out-of-sync detector determines an out-of-sync processing set by majority voting.

41. A method according to Claim 31, comprising the steps of, in response to the identification of an out-of-sync processing set, selecting one of the remaining two processing sets, supplying an interrupt to the out-of-sync processing set and the remaining processing set to cause said out-of-sync and remaining processing sets to idle, reintegrating one of said out-of-sync and remaining processing sets while maintaining a software log of memory write events, and then reintegrating said other of said out-of-sync and remaining processing sets using said software log.

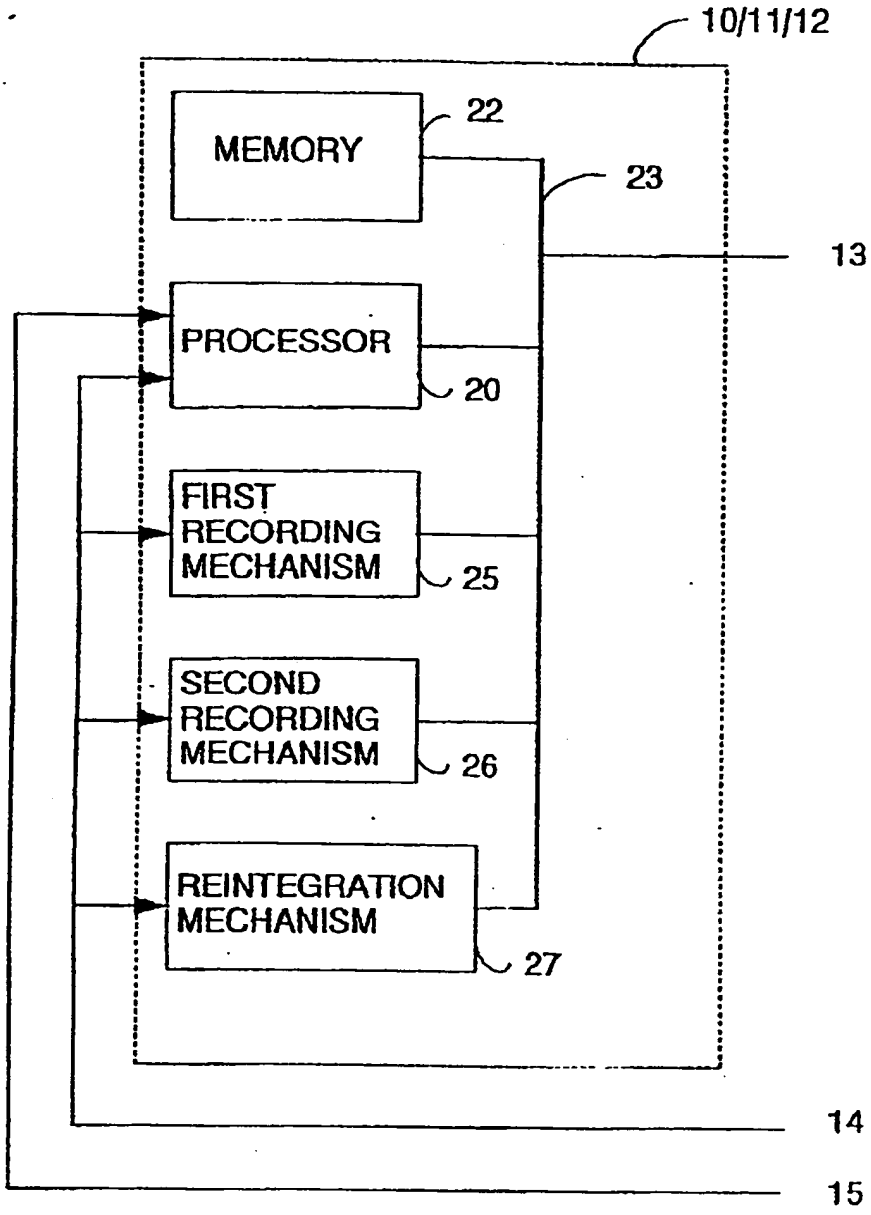
【FIG.1】


PROCESSOR SET 10
PROCESSOR SET 11
PROCESSOR SET 12
VOTER
I/O
CLK
13
14
15
16
17
18

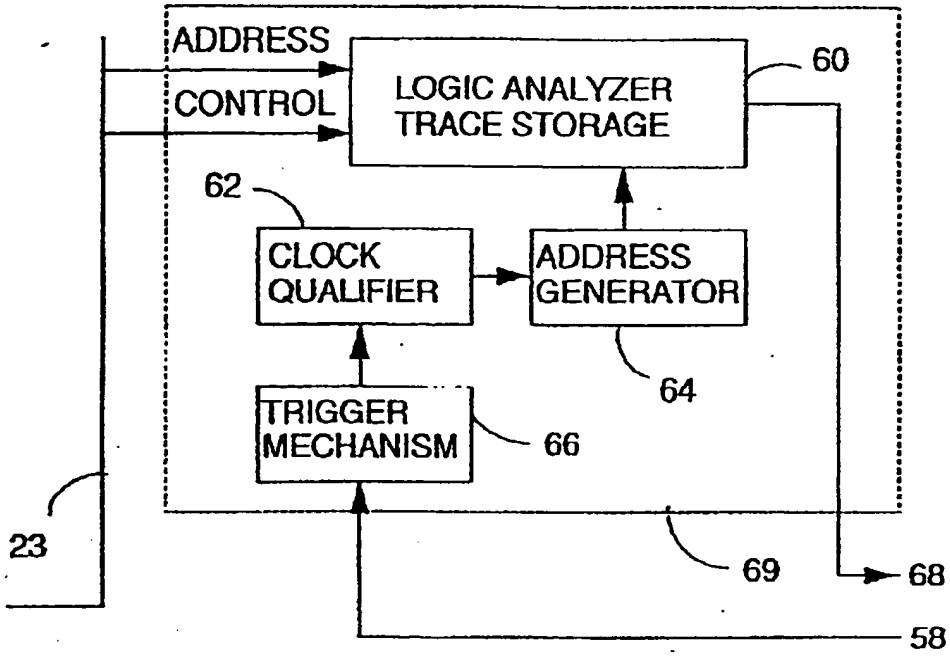
【FIG.2】


MEMORY
PROCESSOR
22
23
20
13
14
15

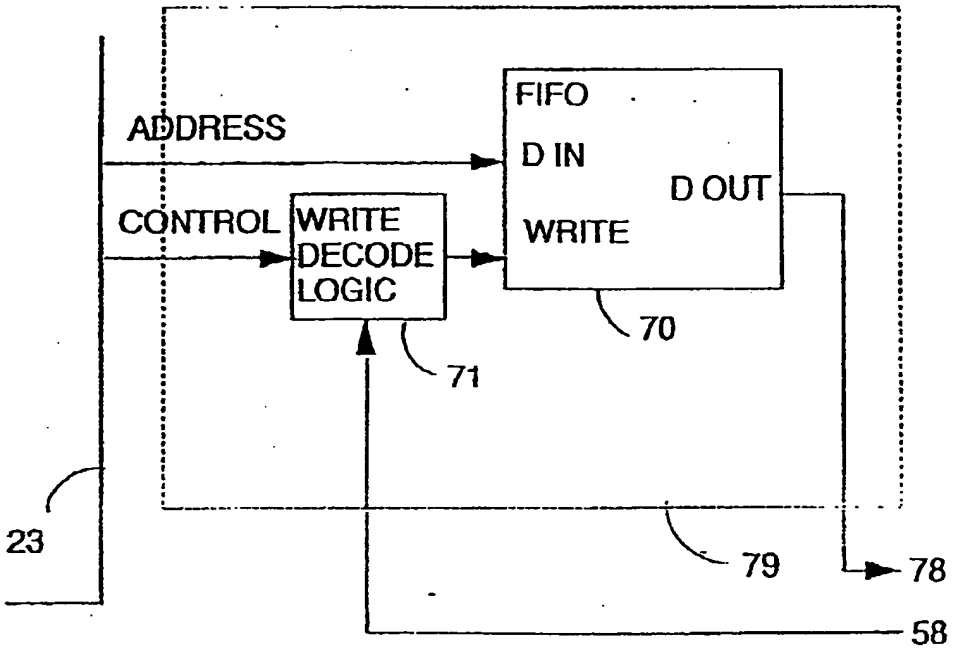
【FIG. 3】


10/11/12
MEMORY
22
23
13
PROCESSOR
20
FIRST RECORDING MECHANISM
25
SECOND RECORDING MECHANISM
26
REINTEGRATION MECHANISM
27
14
15

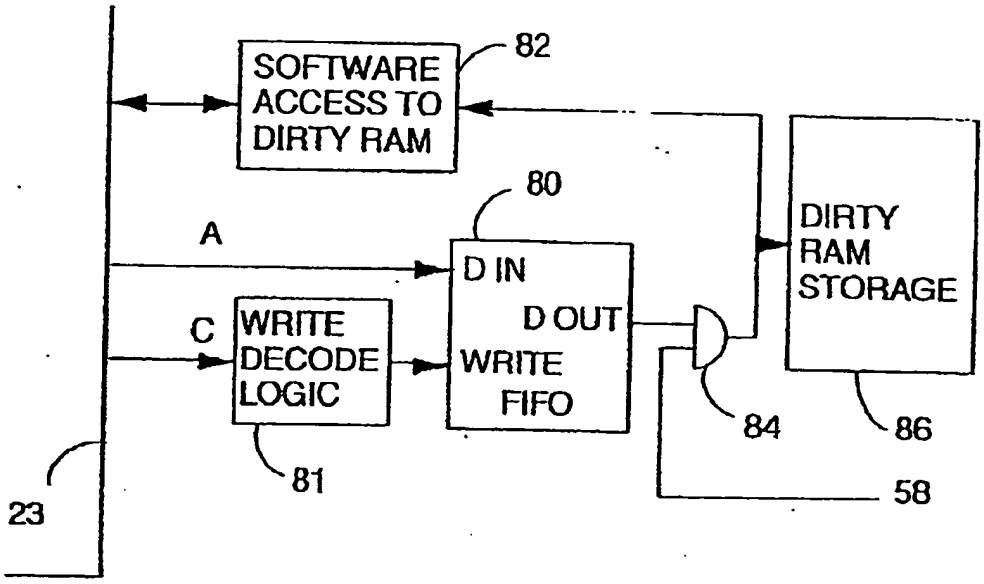
【FIG.6】


ADDRESS
CONTROL
LOGIC ANALYZER
TRACE STORAGE
60
62
CLOCK
QUALIFIER
ADDRESS
GENERATOR
64
TRIGGER
MECHANISM
66
23
69
68
58

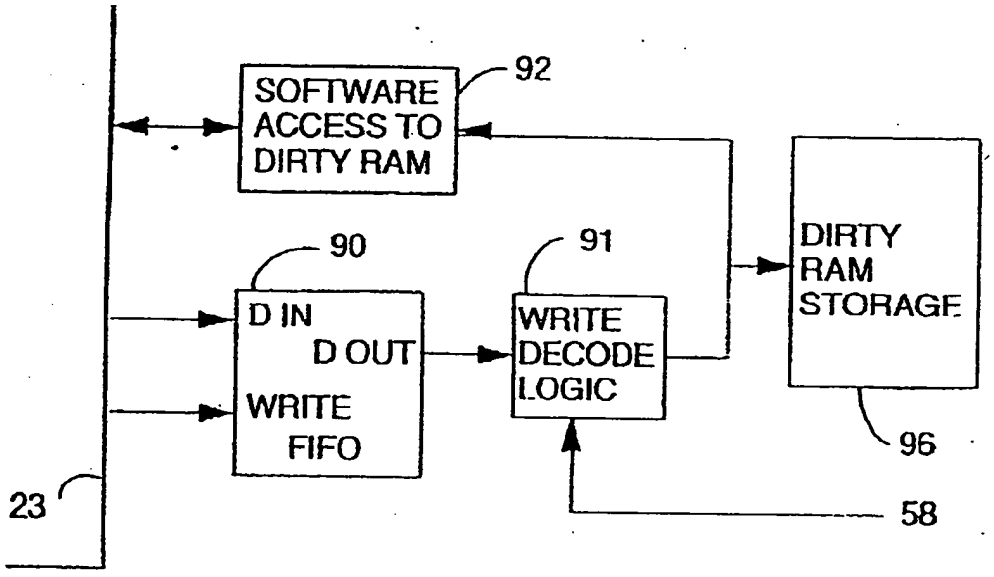
【FIG.7】


FIFO
D IN
D OUT
WRITE
ADDRESS
CONTROL
WRITE
DECODE
LOGIC
70
71
23
79
78
58

[FIG. 8]


SOFTWARE ACCESS TO DIRTY RAM
82
80
A
D IN
D OUT
WRITE FIFO
C
WRITE DECODE LOGIC
81
23
DIRTY RAM STORAGE
84
86
58

[FIG. 9]


SOFTWARE ACCESS TO DIRTY RAM
92
90
91
D IN
D OUT
WRITE FIFO
WRITE DECODE LOGIC
DIRTY RAM STORAGE
96
23
58

ABSTRACT OF THE DISCLOSURE

A memory management system for a fault tolerant computer system includes a first recording mechanism (25) which can be activated to record memory update (write) events, a second recording mechanism (26) having a capacity to record at least a limited number of memory update events, a fault input for a fault signal to activate the first recording mechanism in the event of a fault (out-of-sync) event and a memory reintegration mechanism (27) to reintegrate at least parts of memory identified in the first and second recording mechanisms. The recording of memory updates is preferably done on a page basis using a dirty ram and secondary dirty page record. Recovery from a minor out-of-sync event between processing sets in a lockstep system can be achieved rapidly and efficiently by copying memory pages identified in the first and second recording mechanisms from a running to an out-of-sync processing set as only a relatively small number of locations within the memory system of either the out-of-sync or the running processing set system will have been modified.